Titre: Title:	Analyse de sécurité des applications d'authentification par NFC
Auteur: Author:	Sabrine Jdaida
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Jdaida, S. (2016). Analyse de sécurité des applications d'authentification par NFC [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2149/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2149/
Directeurs de recherche: Advisors:	Jean-Marc Robert, & Jose Manuel Fernandez
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

ANALYSE DE SÉCURITÉ DES APPLICATIONS D'AUTHENTIFICATION PAR NFC

SABRINE JDAIDA
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
MAI 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ANALYSE DE SÉCURITÉ DES APPLICATIONS D'AUTHENTIFICATION PAR NFC

présenté par : JDAIDA Sabrine
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme NICOLESCU Gabriela, Doctorat, présidente
M. FERNANDEZ José M., Ph.D., membre et directeur de recherche
M. ROBERT Jean-Marc, Ph.D., membre et codirecteur de recherche
M. LANGLOIS J.M. Pierre, Ph.D., membre

À mes parents
À mon frère et mes soeurs
À tous mes ami(e)s ...

Qui n'aime pas gravir la montagne,
vivra éternellement au fond des vallées ...
– Abou El Kacem Chebbi.

REMERCIEMENTS

Je dédie ce mémoire de maîtrise à toutes les personnes qui ont contribué de près ou de loin à la réalisation de mon projet de recherche. Les travaux présentés dans ce document ont été réalisés au laboratoire de recherche en sécurité des systèmes d'information (SecSI). Je tiens tout d'abord à remercier mon directeur de recherche, Monsieur José M. Fernandez, pour m'avoir guidée, encouragée pour élaborer cette maîtrise. Ça faisait vraiment un grand plaisir de travailler avec lui pendant ces trois années. Il était toujours disponible pour me soutenir moralement, scientifiquement ainsi que financièrement. De plus, ses précieux conseils m'étaient indispensables pour la réalisation de ce projet.

Ces mots de remerciements s'adressent également à mon codirecteur Monsieur Jean-Marc Robert, professeur à l'École de technologie supérieure (ETS) de Montréal, pour ses recommandations qui ont significativement contribué à l'avancement de mes travaux de recherche et d'avoir accepté de corriger mon mémoire.

Je désire grandement remercier madame Gabriela Nicolescu et monsieur J.M. Pierre Langlois pour avoir accepté d'examiner mon travail, pour leurs remarques et leurs recommandations qui ont permis d'améliorer ce mémoire. Ils m'ont fait l'honneur de participer à mon Jury.

Je tiens à remercier aussi la Mission Universitaire de Tunisie en Amérique du Nord (MUTAN) et le Ministère de l'Éducation, du Loisir et du Sport (MELS) de m'avoir accordé une bourse d'exemption aux frais de scolarité majorés.

J'aimerais remercier tous les étudiants que j'ai pu rencontrer sous la direction du professeur Fernandez : Fanny, Simon, François, Etienne et Antoine dont le côtoiement m'a considérablement aidé dans l'avancement de mes travaux ainsi que dans mon développement personnel. Ils ont contribué à former une ambiance de travail amicale et chaleureuse.

RÉSUMÉ

Les téléphones portables nouvelle génération intègrent des puces de communication à champ raproché (NFC). Grâce à l'émergence rapide de cette technologie, ces téléphones sont capables de jouer le rôle de cartes de crédit, cartes d'accès ou de tickets de transport. Ces applications mobile dites sans contact suscitent un intérêt grandissant. Qu'il s'agisse de contrôle d'accès physique, de transport en commun ou de paiement mobile, la sécurité est toujours un enjeux. En effet, plusieurs solutions d'authentification NFC présentent des risques en termes d'intégrité des données ce qui les rend vulnérables à différentes attaques notamment le clonage des cartes.

Ce mémoire se place dans le cadre d'étude sur la sécurité des applications NFC sur mobile. Le domaine de la sécurité comporte plusieurs axes, tels que : la confidentialité, la détection d'intrusion, l'authentification, le contrôle d'accès, la protection de la vie privée, etc. Dans le cadre de ce projet de recherche, nous nous focalisons sur la problématique d'authentification.

Malheureusement, plusieurs solutions d'authentification, basées soit sur la technologie de carte sans contact d'identification par radio fréquence (RFID) soit sur la technologie NFC, ont démontrées d'importantes failles de sécurité, surtout à cause de la faiblesses des algorithmes cryptographiques employés. Cependant, le recours aux plateformes mobiles rend possible l'utilisation de protocoles cryptographiques à clé publique. Nous proposons une solution d'authentification basée sur une infrastructure à clé publique (ICP) en utilisant la cryptographie à courbe elliptique (ECC). Nous avons construit un prototype de preuve de concept de cette solution, que nous avons utilisé pour vérifier sa viabilité en termes de temps d'exécution.

Une autre des problématiques des applications NFC sur mobile est le stockage des données sensibles notamment les clés cryptographiques. Nous avons donc exploré dans ce mémoire les différentes solutions de stockage de clés cryptographiques. Plusieurs solutions fondamentalement matérielles sont utilisées constituant un environnement isolé de stockage et d'exécution appelés *Secure Element* (SE). Nous avons donc étudié les solutions de SE disponibles, leur complexité et leur disponibilité sur la plateforme mobile Android ainsi que la possibilité d'une implémentation logicielle de SE, afin d'établir les avantages et désavantages de chacune en termes de sécurité.

ABSTRACT

The newer generations of smartphone integrate Near-field Communication (NFC) chips. With the fast emergence of this technology, mobile phones are able to play the role of credit cards, access control badges or transit passes. These contactless mobile applications have generated growing attention in the past few years. From access control solutions to contactless mobile payment, security is always an issue. Many existing NFC authentication solutions have risks in terms of data integrity that make them vulnerable to card cloning.

This thesis studies the security of NFC mobile applications. Security has multiple axes, such as confidentiality, intrusion detection, authentication, access control, privacy, etc. In the context of this research project, we focus on the security of authentication.

Unfortunately, several authentication solutions based on Radio Frequency Identification (RFID) contactless card technology or NFC have shown important security vulnerabilities, especially due to the use of weak cryptographic algorithms. Nonetheless, the use of mobile platforms makes possible the use of public-key cryptographic protocols. We propose an authentication solution based on a Public-Key Infrastructure (PKI) employing Elliptic Curve Cryptography (ECC). We have built a proof-of-concept prototype of this solution that we have used to verify its viability in terms of execution time.

Another issue with mobile NFC applications is the storage of sensitive data, in particular cryptographic keys. We have explored in this thesis different cryptographic key storage solutions. Several solutions are essentially hardware based and use an isolated execution and storage environment called the Secure Element (SE). We have studied the various SE options available on the Android platform, including the option of software SE, in order to establish the advantages and disadvantages of each of these options in terms of security.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AA authentification active (*Active Authentication*)

AES *Advanced Encryption Standard*

APDU *Application Protocol Data Unit*

ARQC *Authorization Request Cryptogram*

BAC *Basic Access Control* (Contrôle d'accès de base)

CSN *Chip Serial Number*

CVM *Cardholder Verification Method*

DDA *Dynamic Data Authentication*

DSA *Digital Signature Algorithm*

DSS *Digital Signature Standard*

EAC *Extended Access Control* (Contrôle d'accès étendu)

ECC cryptographie à courbe elliptique (*Elliptic Curve Cryptography*)

ECDH *Elliptic Curve Diffie-Hellman*

ECDSA *Elliptic Curve Digital Signature Algorithm*

EMV *Europay Mastercard Visa*

FIPS *Federal Information Processing Standard*

HCE *Host Card Emulation* (Mode d'émulation de carte)

ICP infrastructure à clé publique (*Public-key Infastructure*, PKI)

HCI interface de contrôleur hôte (*Host Controller Interface*)

HMI interface homme machine (*Human-Machine Interface*)

JNI *Java native interface*

LLCP *Logical Link Control Protocol*

MAC *Mandatory Access Control*

MRZ *Machine Readable Zone* (Zone de lecture automatique)

MVD machine virtuelle Dalvik

NDEF *NFC Data Exchange Format*

NDK *Native Development Kit*

NFC communication à champ raproché (*Near-Field Communication*)

NIP numéro d'identification personnel (*Personal Identification Number*, PIN)

NIST *National Institute of Standards and Technology* (Institut national des normes et de la technologie)

NSA *National Security Agency*

OSDP *Open Supervised Device Protocol*

OTA *Over The Air* (mise à jour "par le réseau")

QSEE *Qualcom Secure Execution Environment*

RBC Banque Royale du Canada (*Royal Bank of Canada*)

RFID identification par radio fréquence (*Radio Frequency IDentification*)

SDA *Static Data Authentication*

SE *Secure Element* (élement sécurisé)

SIM *Subscriber Identity Module*

SNEP *Simple NDEF Exchange Protocol*

SoC *System on Chip*

SSL *Secure Sockets Layer*

SWP *Single Wire Protocol*

TEE *Trusted Execution Environment* Environnement d'exécution fiabilisé

TC *Transaction Certificate*

TSM *Trusted Service Manager*

UID identifiant unique (*Unique IDentifier*)

CHAPITRE 1

INTRODUCTION

1.1 Problématique de l'authentification basée sur RFID

La technologie d'identification par radio fréquence (RFID) permet à un élément émettant un champ électromagnétique de détecter et identifier un transpondeur qui se trouve dans son champ proche. L'utilisation de cette technologie a connu un essor fulgurant dans les dernières décennies. Le paradigme d'utilisation le plus fréquent est celui où l'émetteur, appelé dans ce cas un *lecteur*, est relié à une infrastructure fixe et détecte et identifie un jeton, appelé *tag* en anglais, qui est doté d'une puce capable de moduler le signal du lecteur de façon unique, lui permettant ainsi de l'identifier.

Les premières applications de cette technologie visaient l'identification de ces tags, par exemple à des fins de contrôle de stock, identification de bétail, etc. Cependant, la possibilité d'identifier ces jetons a vite donné lieu à des applications d'*authentification*, où sur la base de cette identification le lecteur donne accès au jeton, ou plutôt au porteur du jeton, à une ressource quelconque. Ainsi, aujourd'hui on constate une utilisation très répandue et croissante de cartes RFID (jeton en format de carte) dans des applications de paiement électronique, transport en commun et contrôle d'accès physique, entre autres. Dans ces systèmes, le lecteur est généralement relié à une infrastructure fixe. Après l'identification du tag par le lecteur, celui-ci retransmet cette information à un système de contrôle d'accès (appelé le *authentification backbone* ou tout simplement le *backbone*), qui prend alors la décision de donner accès au porteur du jeton ou non.

Cependant, il y a une différence importante entre des applications d'identification versus des applications d'authentification. En principe, le système qui contrôle l'accès à la ressource en question —p.ex. le système de paiement, le guichet de transport en commun, le système de contrôle d'accès physique— a besoin de savoir qui est le porteur du jeton, donc de l'identifier, afin de savoir s'il peut lui donner accès ; c'est-à-dire pour déterminer ses droits d'accès, ce qu'on appelle *autorisation*. Une technologie d'identification, telle la RFID, s'avère donc utile et même nécessaire dans des applications d'authentification. Or, afin que cette utilisation soit sécuritaire, il faut que ces systèmes d'identification aient une propriété supplémentaire : l'impossibilité pour quelqu'un d'autre que le porteur du jeton d'obtenir accès au système en se faisant passer par le porteur du jeton. Cet objectif général de sécurité que doit remplir tout système d'authentification par "quelque chose qu'on a" (p.ex. une clé, un jeton RFID,

une carte, etc.) se traduit par plusieurs propriétés de sécurité. L'une des plus importantes est la *non-clonabilité* des jetons, c'est-à-dire qu'il faut qu'il soit difficile pour une personne non autorisée de produire une copie d'un jeton valable pour se servir ultérieurement de cette copie pour obtenir accès à la ressource. Cette propriété est issue d'un modèle d'attaque simple et courant où l'attaquant gagne l'accès au jeton à l'insu de son propriétaire pendant un laps de temps limité, par exemple [1].

Malheureusement, les systèmes RFID n'exhibent pas cette propriété de non-clonabilité de façon intrinsèque. Effectivement, ils n'ont pas été conçus à l'origine pour ces fins, mais seulement pour l'identification, en l'absence d'un attaquant malveillant. Cette faiblesse a été exhibée et exploitée dans le cas des cartes RFID dites de première génération aussi appelées cartes de proximité. Celles-ci transmettent un identifiant unique (UID) de 64 bits en modulant l'onde porteuse émise par le lecteur. Dans des systèmes de contrôle d'accès physique, par exemple, l'UID est lu et retransmis par le lecteur à un contrôleur d'accès qui décide alors si le porteur de ce jeton (identifié par le code à 64 bits) doit gagner accès à la ressource. Il a été démontré qu'avec très peu de ressources matérielles il est possible de construire un cloneur de carte qui peut obtenir l'UID en se faisant passer par un lecteur. Cette information peut alors être réécrite sur une carte RFID vierge ou retransmise par le cloneur de carte pour émuler la carte originale et ainsi gagner accès à la ressource [2].

Ainsi, une deuxième génération de cartes dites semi-actives a été introduite ultérieurement pour contrer cette faiblesse. D'une part, ces cartes contiennent une puce capable de réaliser des calculs cryptographiques simples. Ainsi, plutôt que de juste transmettre l'UID, ces cartes sont capables d'implémenter de vrais protocoles d'authentification à défi-réponse, où l'information transmise par la carte est différente à chaque fois. Avec l'utilisation de ces protocoles, il est beaucoup plus difficile pour l'attaquant de reproduire le comportement de la carte, car en principe le défi ou "question" émise par le lecteur est différente à chaque fois, et autant en sera la réponse. L'utilisation de fonctions cryptographiques sécuritaires garantit que l'attaquant ne puisse pas facilement trouver les réponses adéquates au défi, en autant que l'attaquant ne connaisse pas les clés cryptographiques secrètes stockées sur la carte et/ou le lecteur. D'autre part, l'UID qui constitue une information "sensible" car elle est utilisée telle quelle comme identifiant dans certains infrastructures d'authentification (par exemple les systèmes de contrôle d'accès physique), peut être protégée par l'utilisation de ces mêmes protocoles cryptographiques. Ainsi, avant de transmettre l'UID, la carte peut vérifier qu'elle interagit

1. Il est important de noter qu'il ne s'agit pas de la seule propriété de sécurité qu'un tel système doit avoir. D'autres propriétés sont nécessaires lorsqu'on considère d'autres modèles attaques, comme par exemple les attaques de *man-in-the-middle*.

2. Il existe même des dispositifs spécialisés disponibles sur le marché, tel le Proxmark III, qui permettent de réaliser facilement ces attaques.

avec un lecteur autorisé en émettant elle aussi un défi et en en vérifiant la réponse donnée par le lecteur. On parle donc ici de protocole d'*authentification mutuelle*.

Regrettablement, plusieurs solutions commerciales de deuxième génération se sont avérées ne pas être sécuritaires. La première raison est que les fonctions cryptographiques utilisées n'étaient pas sécuritaires. Il s'agissait de fonctions propriétaires dont les détails n'avaient pas été publiés et donc n'avaient pas fait l'objet d'examen approfondi par la communauté internationale d'experts en cryptographie. Par exemple, dans le cas des cartes HID iClass (utilisées dans les systèmes de contrôle d'accès physique) lorsque les détails sur les fonctions cryptographiques ont pu être découverts, rapidement des protocoles de cryptanalyse ont été trouvés qui permettaient de reconstruite la clé cryptographique secrète, et donc de cloner des cartes à volonté. La deuxième raison est que ces protocoles d'authentification utilisaient des fonctions cryptographiques à clé symétrique, c'est-à-dire que c'est la même clé secrète qui est utilisée pour calculer la réponse et pour vérifier qu'il s'agit de la bonne réponse. Ceci peut amener à des vulnérabilités si les lecteurs contenant ces clés sont compromis. Dans l'exemple des cartes iClass, une attaque a été mise en œuvre où en piratant des lecteurs, des chercheurs ont réussi à obtenir la clé cryptographique maître qui y était stockée, leur permettant ainsi de cloner des cartes à volonté ; et ce sans avoir à recourir à un protocole de cryptanalyse exploitant les faiblesses cryptographiques.

Ainsi, nous pouvons tirer deux leçons apprises fondamentales quant à l'utilisation des technologies RFID à des fins d'authentification.

Premièrement, il est nécessaire d'utiliser des fonctions cryptographiques sécuritaires connues, qui ont étés examinées et reconnues à cet effet par la communauté scientifique. En d'autres mots, il n'est pas prudent de se baser sur le principe de "sécurité par obscurité" comme ça été le cas, mais plutôt sur le principe de Kerckhoffs, où la sécurité se base exclusivement sur le secret de la clé cryptographique. Ce principe qui fait partie des meilleurs pratiques en sécurité, adopté aujourd'hui presque partout dans les applications sécurisées telles les communications et les paiements bancaires par Internet, n'est en fait pas si facile à appliquer dans les systèmes RFID. Effectivement, les cartes RFID ne disposent pas de source d'alimentation indépendante et doivent alimenter leurs circuits électroniques à même le champ électromagnétique émis par le lecteur. Ceci limite significativement la taille des circuits et donc des calculs qui peuvent y être réalisés. C'est une des raisons qui explique partiellement pourquoi de si faibles fonctions cryptographiques se trouvent sur les cartes RFID de deuxième génération. Ainsi, l'idéal serait de recourir à une technologie alternative permettant de réaliser des calculs cryptographiques plus importants tels que des fonctions reconnues soit à clé symétrique, telles que AES, soit à clé publique ou asymétrique, telles que RSA ou El-Gamal. À cet effet, l'utilisation de plateformes mobiles, telles que des tablettes

ou des téléphones intelligents, s'avère prometteuse. D'une part, l'ubiquité de ces plateformes devenues maintenant très accessibles rend viable leur utilisation comme jetons d'authentification dans plusieurs des applications où les cartes RFID étaient utilisées à cet effet. C'est d'ailleurs pour cette raison que les manufacturiers de ces plateformes ont développé et déployé dans leurs nouveaux téléphones la technologie communication à champ raproché (NFC), qui leur permet de communiquer avec des lecteurs de cartes RFID. Cependant, il est décevant de constater que la plupart des applications NFC déployées jusqu'à présent sont vulnérables aux mêmes types d'attaques que leurs prédécesseurs NFC, entre autres étant donné l'utilisation des mêmes faibles fonctions cryptographiques.

Deuxièmement, le fait que des fonctions cryptographiques symétriques (à clé privée) soient utilisées représente une faiblesse si les clés cryptographiques correspondantes ne peuvent pas être suffisamment sécurisées, non seulement sur les cartes mais aussi sur les lecteurs. La technologie actuelle de cartes RFID (comme celles des cartes à puce à contact ou *smartcards*) fournit un niveau de protection adéquat contre la plupart des attaquants ; il est en effet assez difficile d'en extraire la clé cryptographique. Il n'en est pas de même pour la plupart des lecteurs de cartes présentement sur le marché. Effectivement, un accès physique non interrompu à ces dispositifs permet d'en extraire la clé assez facilement. Une façon évidente de faire face à ce risque est donc de concevoir et déployer des lecteurs mieux sécurisés en combinant une architecture matérielle et logicielle adéquate à ces fins. Ceci est cependant en dehors des objectifs de ce travail de recherche. Une autre façon de minimiser ce risque est d' utiliser la cryptographie à clé publique, dont les propriétés d'asymétrie font en sorte que la clé cryptographique de vérification soit différente de la clé utilisée pour générer les réponses. En conséquence, la clé de vérification n'a plus besoin d'être confidentielle et peut même être rendue publique, d'où le nom. Cependant, afin d'éviter que le lecteur/vérificateur ait à stocker les clés publiques de tout potentiel jeton voulant s'authentifier, il est possible de recourir à une infrastructure à clé publique (ICP). L'utilisation d'algorithme à clé publique et des ICP est très répandue dans les applications Web et le commerce électronique, et donc il est naturel d'y penser comme solution potentielle pour l'authentification par RFID ou NFC.

1.2 Objectifs de recherche

Comme nous l'avons mentionné ci-haut, l'utilisation de la technologie RFID à des fins d'authentification reste problématique. Il n'en est pas moins pour sa technologie sœur du NFC sur les plateformes mobiles, ou du moins dans son incarnation actuelle.

Cependant, et tel que nous l'avons identifié, l'utilisation de plateformes mobiles rend possible l'utilisation de protocoles cryptographiques à clé publique, ce qui de concert avec

l'utilisation d'ICP, devrait réduire significativement les risques d'attaques contre ces systèmes. Or, cette approche n'est pas celle qui est suivie par les manufacturiers de lecteurs et cartes, ni par les manufacturiers de plateformes mobiles, ni par les développeurs d'application authentification NFC. Ceci nous mène donc à nous poser des questions sur les obstacles qui s'opposent à l'adoption d'une telle solution dans ce contexte. L'obstacle principal à l'adoption de cryptographie à clé publique a typiquement été relié à la performance. Effectivement, lors de son introduction dans les années 80, la puissance de calcul des ordinateurs de l'époque et des contraintes de bande passante avaient plutôt forcé l'adoption de protocoles mixtes combinant les algorithmes à clé privée avec ceux à clé publique, comme dans le cas du protocole *Secure Sockets Layer* (SSL).

D'autre part, l'utilisation des plateformes mobiles comme jetons d'authentification, combinée ou non avec l'utilisation de cryptographie à clé publique ne constitue pas une panacée résolvant toutes les problématiques de sécurité. Quelque soit la technologie cryptographique utilisée, la confidentialité des clés utilisées pour répondre correctement lors de protocoles d'authentification reste la pierre angulaire de la sécurité de ces protocoles, tel que l'indique le principe de Kerckhoffs. Or, la même ubiquité des plateformes mobiles, qui rend possible leur utilisation comme jetons, ainsi que leur versatilité, constitue aussi une menace en soi. Effectivement, tel que nous l'avons vu dans les plateformes traditionnelles (ordinateurs de bureau et ordinateurs portables), les pirates et cybercriminels ont développé des outils et des méthodes très variés et efficaces pour pénétrer dans ces plateformes et en soutirer des informations confidentielles qui y sont stockées ou y transigent, tels que des mots de passe, des numéros de carte de crédits, etc. Nous constatons également une montée du phénomène du logiciel malveillant sur les plateformes mobiles, à différentes fins de monétisation des plateformes mobiles par des cybercriminels. Il est donc naturel d'espérer qu'au fur et à mesure que les plateformes mobiles sont utilisées comme jetons d'authentification, une nouvelle génération d'attaques visant à subtiliser les clés cryptographiques correspondantes pourra voir le jour. Il faut donc se poser la question sur les mécanismes envisageables pouvant les contrecarrer.

L'objectif principal de ce travail est donc d'établir la viabilité technologique et pratique de l'utilisation d'algorithmes à clé publique et d'ICP dans des applications d'authentification par NFC sur des plateformes mobiles. Afin d'atteindre cet objectif il nous faut cependant répondre aux questions de recherche suivantes :

1. **Rapidité des transactions d'authentification**. Est-il possible de réaliser des transactions d'authentification utilisant de la cryptographie à clé publique entre une plateforme mobile et un lecteur NFC ou RFID dans un laps de temps comparable à celui des transactions d'authentification RFID actuelles ? Plus concrètement, un des avantages significatifs de l'aspect "sans contact" du RFID est la convivialité liée à la rapidité de la

procédure d'authentification, plus particulièrement du fait qu'il est plus facile et rapide pour l'usager d'approcher une carte RFID d'un lecteur, plutôt que d'insérer une carte, comme dans le cas d'une carte à puce. Est-ce que cet avantage est maintenu dans le contexte d'authentification basée sur la cryptographie à clé publique ?

2. **Protection des clés cryptographiques**. Quels sont les scénarios d'attaques sur des plateformes mobiles qui pourraient mettre en cause la confidentialité des clés cryptographiques utilisées pour l'authentification ? Plus concrètement, étant données les perspectives de compromission de l'intégrité de la plateforme mobile, par exemple par un logiciel malveillant, quelles sont les méthodes pouvant être utilisées pour protéger les clés cryptographiques ? Quels sont les avantages et désavantages de ces options en termes d'accessibilité pour les développeurs d'applications, de sécurité et de convivialité ?

La technologie NFC n'étant pas encore complètement mature, nous avons été quelque peu limités dans les moyens disponibles pour répondre à ces questions. Premièrement, il n'existe pas de lecteur RFID reconfigurable sur lequel nous aurions pu facilement implémenter un protocole d'authentification basé sur les algorithmes à clé publique. Nous avons donc développé et implémenté un prototype équivalent qui fonctionne entre deux plateformes mobiles, où l'une d'elles joue le rôle de "lecteur" alors que l'autre joue le rôle de "carte" ou jeton. Les tests de performance réalisés sur cette plateforme nous ont permis de répondre à la première question.

Pour répondre à la deuxième question, nous avons recherché et évalué les différentes solutions proposées par l'industrie pour sécuriser les clés, notamment le concept de *Secure Element* (SE) mis de l'avant par les manufacturiers de plateformes mobiles et les opérateurs de télécommunications. Cependant, nous avons été sévèrement limités dans l'évaluation en laboratoire de ces solutions étant données les importantes restrictions imposées par ces manufacturiers à l'accès des composantes critiques de cette approche, notamment les puces ou carte à puces qui implémente le SE.

1.3 Plan du mémoire

Le reste du mémoire est divisé en 4 chapitres. Au Chapitre 2, nous étudions tout d'abord les différentes attaques connues sur les systèmes d'authentification RFID.

Dans le chapitre suivant, nous tâchons de répondre à la deuxième question en étudiant les mécanismes de sécurité pouvant contribuer à protéger les clés cryptographiques sur une plateforme mobile Android. Nous décrivons en premier lieu le modèle de sécurité intrinsèque du système d'opération Android. Puis, nous nous sommes intéressés à l'évaluation compara-

tive des différentes solutions de stockage de clé cryptographique sur Android, notamment la notion de SE.

Pour répondre à la première question, nous proposons dans le chapitre 5 différentes possibilités d'implémentation de protocoles d'authentification basés sur la cryptographique à clé publique. Nous décrivons l'implémentation que nous en avons faites dans une application d'authentification testé entre deux plateformes Android. Nous décrivons aussi les résultats des tests de performance que nous avons réalisés sur ces plateformes.

Finalement, nous clôturons ce mémoire en récapitulant sur les résultats obtenus, leurs limitations, en discutant des implications de nos résultats sur la viabilité de l'utilisation de telles solutions dans un déploiement réel d'une telle solution d'authentification.

CHAPITRE 2

DE L'IDENTIFICATION RADIO FRÉQUENCE A LA COMMUNICATION EN CHAMP PROCHE : LA SÉCURITÉ EN QUESTION

Ce chapitre est une introduction aux technologies RFID et NFC dans un contexte de sécurisation du processus d'authentification. Nous y présentons des notions de base relatives au fonctionnement de ces deux technologies, des attaques connues sur les systèmes RFID ainsi qu'une étude de fiabilité de ces attaques sur un système NFC.

2.1 Identification par radio fréquence (RFID)

L'utilisation de la RFID se développe rapidement dans de nombreuses industries différentes. Comme nous l'avons mentionné au Chapitre 1, cette technologie est utilisée non seulement dans des applications traditionnelles telles que le suivi des stocks, mais aussi dans des services de sécurité tels que les passeports électroniques, le contrôle d'accès physique et les cartes bancaires RFID qui nécessitent des mécanismes d'authentification assurant un haut niveau de sécurité. Pour pouvoir analyser ces mécanismes d'authentification RFID, nous commençons par présenter cette technologie. En effet, un système RFID se compose d'au moins deux éléments : un lecteur et une étiquette (figure 2.1).

Une étiquette RFID, également nommé jeton ou *tag* en anglais, est un support d'identification et de localisation électronique. Elle dispose d'une puce qui lui permet de stocker les données nécessaires à son identification et de moduler le signal du lecteur. Cette puce est reliée à une antenne qui permet à la carte de transmettre des informations vers un lecteur RFID.

Un lecteur RFID transmet à travers des ondes radio l'énergie nécessaire à l'étiquette. Grâce à cette énergie, l'étiquette située dans ce champ électromagnétique peut répondre aux requêtes envoyées par le lecteur. Un lecteur RFID est composé généralement d'une antenne, d'un circuit électronique et d'une source d'alimentation.

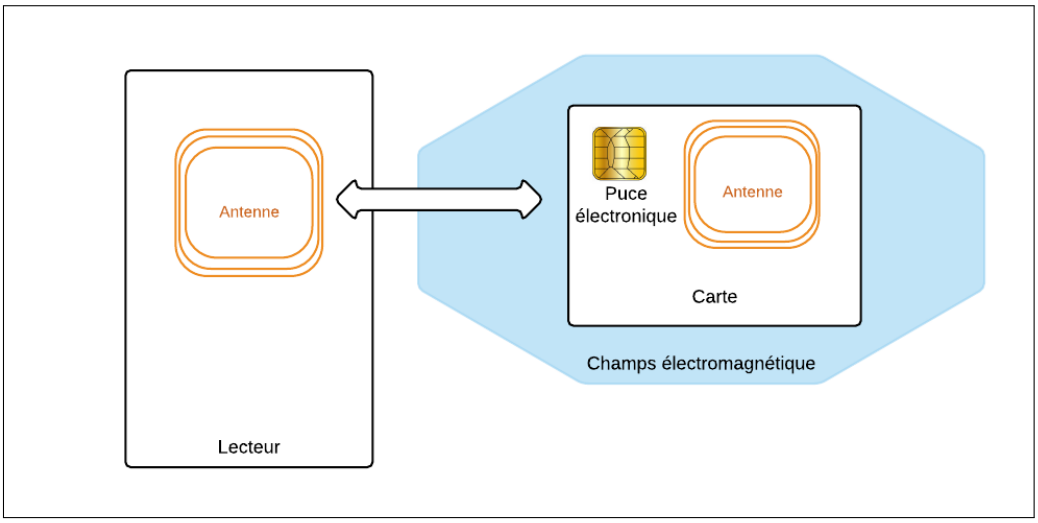

Figure 2.1 Architecture d'un système RFID

2.1.1 Implémentations et applications

Comme la technologie RFID a été bien adoptée pour répondre aux besoins d'authentification, on constate sa croissante intégration dans des applications telles que le contrôle d'accès physique, le paiement électronique et le transport en commun. Nous nous intéressons dans ce qui suit à détailler le fonctionnement de ces applications et les faiblesses de sécurité connues. Nous notons deux type d'étiquettes : actives et passives. Les étiquettes actives sont équipées d'une énergie propre qui leur permet d'émettre un signal de manière autonome et sont généralement utilisées en identification à longue distance. De ce fait, le principal avantage repose sur la longue distance à laquelle elles peuvent communiquer les données sans qu'un lecteur RFID se situe à proximité du jeton. Cependant, le coût assez élevé de ces étiquettes et leur durée de fonctionnement limitée constituent un réel frein à leur développement. Les étiquettes passives sont les plus utilisées sur le marché. Elles ne disposent pas de source d'énergie et utilisent le puissance du signal électromagnétique émis par le lecteur RFID afin d'alimenter les circuits électroniques leur permettant de faire des calculs et d'émettre des données. Dans le cadre de son mémoire de maîtrise, Brun-Murol (2012) a classé ce type d'étiquette selon leur niveau de sécurité :

1. **Étiquettes de 1ère génération**. Elles envoient au lecteur son UID sans aucune authentification. Il suffit donc d'interroger l'étiquette avec la même requête que le lecteur envoie pour obtenir son UID.
2. **Étiquettes de 2ème génération**. Ces étiquettes ont introduit un mécanisme de défi/réponse (*Challenge/Response* en anglais). En effet, le lecteur envoie un *défi* ou

nonce à l'étiquette qui prouve son identité en répondant à ce défi. La plupart des méthodes d'authentification utilisés par l'industrie sont propriétaires. Avec le défi/réponse, il est également possible de réaliser une authentification *mutuelle*, où le lecteur s'identifie préalablement à l'étiquette RFID. Cependant, plusieurs implémentations ne l'utilise pas ou l'implémente de façon déficiente (génération non aléatoire des défis).

Nous notons aussi différentes implémentations de cette technologie avec différents niveau de sécurité.

Mifare Classique. Produit de la société NXP Semiconductors très répandue notamment en contrôle d'accès et transport publique. Il s'agit d'un exemple d'étiquette passive de deuxième génération disposant de mécanisme d'authentification mutuelle permettant d'authentifier à la fois le lecteur et l'étiquette RFID. Ces cartes utilisent un protocole de chiffrement symétrique par flux baptisé « CRYPTO-1 ».

Comme l'indique la figure 2.2, une carte Mifare Classique est divisée en 16 secteurs :

1. Un secteur est composé de 4 blocs de données constituant une taille totale de 64 octets (512 bits). Chaque secteur comporte 3 blocs de données et 1 bloc de sécurité. Les clés et les conditions d'accès contenues dans le bloc de sécurité sont propres à chaque secteur.
2. Un bloc de données comporte 16 octets (128 bits) et constitue le plus petit élément adressable. Nous notons 3 types de blocs, le bloc 0 ou bloc du fabriquant (1er bloc du secteur 0) contenant le UID, des blocs de données et des blocs de sécurité. Ce dernier type de bloc est constitué lui même de deux clés de sécurité de 6 octets, 3 octets de contrôle d'accès et d'un octet de données.

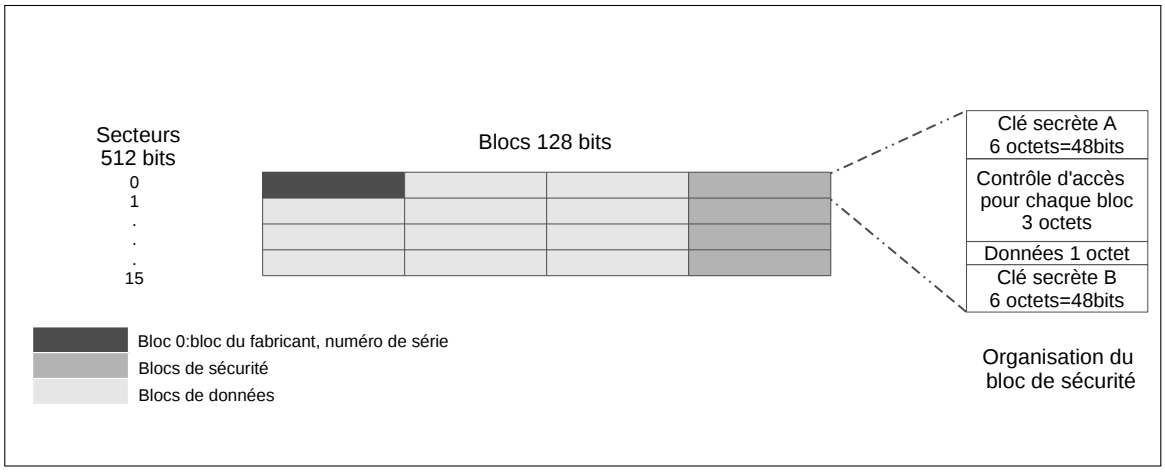

Figure 2.2 Organisation de la mémoire EEPROM d'un jeton RFID de 2ème génération

Comme mentionné, chaque bloc du secteur dispose de ses propres conditions d'accès définies dans le bloc de sécurité. De même chaque secteur est protégé séparément et indépendamment des autres secteurs ce qui implique un mécanisme d'authentification par secteur. Cette authentification nécessite les dispositions suivantes :

1. 2 clés (A et B) stockées dans chaque secteur.
2. 2 clés (A et B) correspondantes stockées dans le lecteur pour chaque secteur.
3. Calcul d'un cryptogramme utilisé pour prouver que le lecteur et la carte détiennent le même secret.
4. La communication entre la carte et le lecteur est chiffrée par la clé de session générée après l'authentification.

Comme discuté précédemment, le recours à la sécurité par obscurité c'est-à-dire l'utilisation d'algorithmes cryptographiques propriétaires et gardés secrets, constitue une mauvaise pratique de sécurité. De ce fait, Karsten Nohl (2007) a réussi à reconstruire l'algorithme utilisé par les cartes Mifare Classique « CRYPTO-1 » et à y identifier différentes vulnérabilités. En effet, la faible taille des clés (48 bits) rend la solution vulnérable à une attaque par force brute, de l'ordre de 10 heures, pour retrouver une clé en utilisant un *FPGA*. De plus, cet algorithme utilise un générateur de nonce prévisible comme les nonces générés ne dépendent que des cycles d'horloge entre le temps où le lecteur a été alimenté et le temps où la génération de nonce est demandée. En exploitant ces vulnérabilités, il est possible de déchiffrer la communication entre le lecteur et la carte, de reconstruire les clés cryptographiques et par la suite de cloner les cartes Mifare Classique.

HID IClass. Produit de la société HID Global, dédié principalement au contrôle d'accès physique. Plusieurs chercheurs scientifiques, dont Milosch Meriac (2010), ont réussi à obtenir la clé maître du produit IClass Standard Security. Cette attaque est possible à cause d'une mauvaise configuration des bits de contrôle d'accès à la mémoire du micro-contrôleur utilisé dans les modèles de lecteurs de cette famille.

Paypass/Paywave. Les compagnies de cartes de crédit ont profité de l'expansion de la technologie RFID pour proposer des cartes munies des technologies dite « sans contact » *Paypass* de Master Card et *Paywave* de Visa. Ces deux technologies visent à faciliter l'expérience d'achat en introduisant la notion de *Tap & Go* qui permet d'effectuer la transaction sans avoir à glisser (*swipe* en anglais) ou insérer une carte de paiement. Nous avons présenté dans l'émission télévisée « JE » diffusée le 15 février 2013 sur la chaîne canadienne TVA, une vulnérabilité de cette technologie. Notre preuve de concept consistait à développer une

application mobile Android qui fonctionne en mode lecteur de carte capable d'interroger différentes cartes de crédit et d'afficher les informations obtenues sur l'écran du mobile. Cette simple application nous a permis d'avoir différentes informations sensibles de la carte bancaire telles que le numéro de la carte, le nom du porteur de carte pour certains types, la date d'expiration ainsi que la liste des dernières transactions effectuées, ce qui constitue une violation flagrante de la vie privée des détenteurs des cartes bancaires sans contact.
En effet, une transaction *Europay Mastercard Visa* (EMV) implique les opérations suivantes :

1. Sélection de l'application : le terminal envoie à la carte une commande SELECT de l'application (Visa, Master Card).
2. La carte répond avec toutes les données de l'application sélectionnée dont le numéro de la carte, la date d'expiration, la date d'expiration et le nom du détenteur de la carte.
3. L'authentification de la carte : le terminal de paiement vérifie l'intégrité des données de la carte selon la méthode d'authentification disponible sur la carte : *Static Data Authentication* (SDA) ou *Dynamic Data Authentication* (DDA). En effet, SDA indique que les données d'application de paiement n'ont pas été manipulées ou modifiées, ce qui rend possible de copier les données de la carte à puce et l'écrire dans une autre carte à puce pour créer une carte contrefaite qui pourrait passer avec succès devant un terminal de paiement. DDA est une forme plus forte de l'authentification de données permettant de contrecarrer les fraudes introduites par le mode SDA en rendant la phase d'authentification *dynamique* impliquant un certificat à clé publique.
4. La vérification du détenteur de la carte, appelée aussi *Cardholder Verification Method* (CVM) selon les options de vérification disponibles : un numéro d'identification personnel (NIP) en mode en ligne, un NIP en mode hors ligne ou pas de vérification.
5. Le choix de type de transaction : en ligne ou hors ligne.
6. La génération du cryptogramme de la transaction selon le type de transaction choisi : *Authorization Request Cryptogram* (ARQC) en mode en ligne qui requiert une autorisation en ligne de l'émetteur de la carte (institution financière) et *Transaction Certificate* (TC) en mode hors ligne.

Nous notons qu'une transaction EMV sans contact diffère peu de celle avec contact. La différence est au niveau de la vérification du détenteur de la carte CVM, cette vérification est annulée dans le cas d'une transaction sans contact. Pour cela, il était nécessaire de limiter les montants pouvant être débités sans présentation de NIP en cas d'utilisation frauduleuse suite à perte ou vol. En 2015, le montant maximum pouvant être débité lors d'un paiement était de 100 $ au Canada et de 20 € en France.

La vulnérabilité que nous avons exploitée était liée au fait que la carte renvoie la totalité des données de l'application sélectionnée avant même d'effectuer l'authentification de la carte. De plus, en 2014, des chercheurs de l'université de Newcastle, Emms *et al.* (2014), ont mis en évidence une faille dans le système qui n'est pas liée à la technique du sans contact à proprement parler, mais plutôt aux applications de paiement. Cette faille consistait à outrepasser les plafonds lors de paiements en devises étrangères en mode hors-ligne sans contrôle du montant. En effet, ces chercheurs ont réussi des transactions *Paywave* en devise étrangère pour des montants allant jusqu'à 999 € sachant qu'il s'agit d'une carte anglaise avec un montant maximum par transaction unitaire de 20 £. En effet, grâce à une modélisation formelle d'une transaction sans contact, ces chercheurs ont constaté qu'il n'était pas possible de modéliser les pré-conditions nécessaires au paiement sans contact lorsqu'il s'agit d'une transaction en devise étrangère.

2.1.2 Attaques RFID connues

La technologie RFID suscite beaucoup d'interrogations en terme de sécurité. Son intégration dans plusieurs services de sécurité tels que le contrôle d'accès physique et le paiement mobile n'a pas été sans difficultés. Comme discuté précédemment, les solutions existantes reposent majoritairement sur des algorithmes cryptographiques symétrique, propriétaires et peu sécuritaires. Ces propriétés ont rendu ces systèmes vulnérables à différentes attaques.

Écoute à distance. Cette attaque est dite passive et consiste à écouter une transaction privée entre un lecteur et une carte dans l'intention d'y déceler des secrets. Un adversaire équipé d'un balayeur radio (*scanner*) pourrait espionner la communication entre deux dispositifs NFC. Généralement, la distance de communication de dispositifs NFC est à proximité ou inférieur à 10 cm. Pour cette raison, une attaque réussie nécessite un bon récepteur de signal de qualité et le décodeur, une antenne qui change de direction en 3 dimensions, et un environnement avec moins de distractions et les obstacles tels que du bruit, de métal ou un mur. De plus, la géométrie de l'antenne et la puissance de l'émetteur (jeton) pourraient également influer sur le succès d'une telle attaque. Les premières attaques d'écoute à distance ont été présentées par Hancke (2006) dans une publication regroupant trois attaques au niveau de la couche physique RFID. En effectuant des tests avec plusieurs tailles d'antennes et plusieurs types d'amplificateurs, Hancke a réussi son attaque avec une distance de lecture maximale de 27 cm.

Un autre facteur non négligeable est le mode de fonctionnement du dispositif émetteur, dans un mode actif le jeton NFC crée son propre champ RF alors qu'en mode passif il utilise le champ RF du lecteur. En effet, dans un mode actif, un dispositif NFC est destiné à chercher

d'autres dispositifs passifs à proximité directe et donc dans ce cas une écoute est possible jusqu'à une distance allant jusqu'à 10 mètres, contrairement à un dispositif en mode passif qui attend silencieusement jusqu'à ce que la puissance active le dispositif. Ainsi, la distance possible de l'écoute clandestine est de 1 mètre. L'écoute à distance pourrait être évitée en utilisant un canal de communication sécurisé.

Activation à distance. Consiste à activer et lire une carte sans le consentement de son propriétaire. Cette attaque est dite active, car l'attaquant doit fournir l'énergie nécessaire au fonctionnement de la carte. Elle permet en général de communiquer avec la carte en dehors de sa plage de fonctionnement.

Attaque par relais. Une attaque relais consiste à établir une communication entre un lecteur et une carte sans contact sans le consentement de son propriétaire. L'attaquant ne fait que relayer mot par mot le message intercepté. L'intégrité de la communication peut être mise en danger et les messages peuvent alors être écoutés, voire modifiés. Une preuve de concept de cette attaque a été implémentée par Markantonakis (2012) et testée dans un contexte de système de paiement en laboratoire où un lecteur et une carte mandataires relaient les messages entre un terminal de paiement et une carte légitimes.

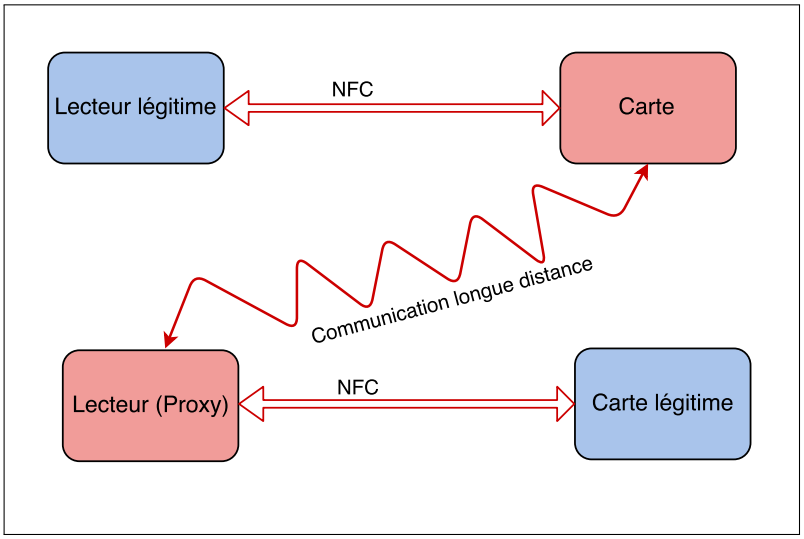

Figure 2.3 Attaque par relais

Attaque de l'homme du milieu. Aussi nommée *Man in the Middle*, cette attaque possède des caractéristiques similaires à l'attaque relais : l'objectif consiste à transmettre les données entre le lecteur et la carte en passant par le relais. Cependant, cette attaque est plus

sophistiquée qu'une attaque par relais. En effet, les bits échangés entre le lecteur et la carte peuvent être modifiés durant le passage dans le relais.

2.1.3 Contre-mesures

Protocole limitateur de distance (*Distance Bounding Protocol*). Le but d'un délimiteur de distance est de permettre au lecteur de s'assurer que le jeton avec lequel il communique se trouve dans un rayon proche de lui. Cette technique permet au lecteur de détecter une attaque par relais ou une attaque de l'homme du milieu. Il consiste en 3 phases d'échanges de bits, lente, rapide puis lente. En mesurant le temps d'aller-retour des échanges, le lecteur décide si l'étiquette est autorisée à continuer la communication avec lui.

Protocole bloquant. L'exemple le plus connu de cette solution est la cage de Faraday, en effet, en utilisant un métal d'une épaisseur inversement proportionnelle à la longueur d'onde du champs électromagnétique émis par le lecteur, on peut bloquer les réponses des étiquettes évitant toute cueillette clandestine d'informations de l'étiquette. Il existe aussi d'autre solutions de protocoles bloquants tels que *RFID Guardian* qui est un dispositif de validation des requêtes émises par le lecteur et bloquant, si nécessaire, toute réponse de la part de l'étiquette interrogée. Pour se faire, la réponse de l'étiquette sera brouillée sélectivement interdisant ainsi son acquisition par le lecteur RFID à l'origine des requêtes.

2.2 Communication à champ rapproché (NFC)

2.2.1 Normes

NFC est une technologie sans contact spécifiée par le NFC Forum en 2004 se basant sur la technologie RFID et conçue spécifiquement pour les dispositifs mobiles. NFC se fonde sur la norme ISO/IEC 14443 du RFID, opérant à la fréquence 13.56 MHz où les appareils peuvent émuler les cartes sans contact de type ISO/IEC 14443, sont en mesure d'alimenter et lire des cartes ISO/IEC 14443, et peuvent aussi échanger des informations en pair à pair. La bande passante requise (le débit des informations) peut-être de 106 kbps, 212 kbps ou 424kbps. Les systèmes NFC diffèrent légèrement des systèmes RFID. En effet, l'architecture d'un système RFID repose sur une structure lecteur/carte alors qu'un système NFC peut jouer le rôle de lecteur et de carte à la fois. Pour spécifier ces modes d'opération, plusieurs normes de communication NFC ont été définies par le NFC Forum comme l'indique la figure 2.4.

– *NFC-IP1.* Constitué des normes ECMA 340 et ISO 18092 qui définissent le mode P2P. Ces normes doivent être compatibles avec les normes ISO 14443-A et Felica (norme Sony appelée aussi type C). A l'heure actuelle, on observe quelques incompatibilités

entre la norme NFC-IP1 et la norme ISO 14443-A. De nombreuses discussions aux comités de normalisation ont lieu pour étudier la convergence de ces deux normes.

– *NFC-IP2,* ou ECMA 352, définit le mode lecteur. La norme impose que les dispositifs NFC soient compatibles en tant que lecteurs de transpondeurs aux standards définis par les normes ISO14443 type A et B et ISO 15693.
– Il n'existe pas encore de norme qui définit le mode d'émulation de carte, mais le NFC Forum, composé de grand acteurs de la RFID, a travaillé sur des spécifications pour ce mode.

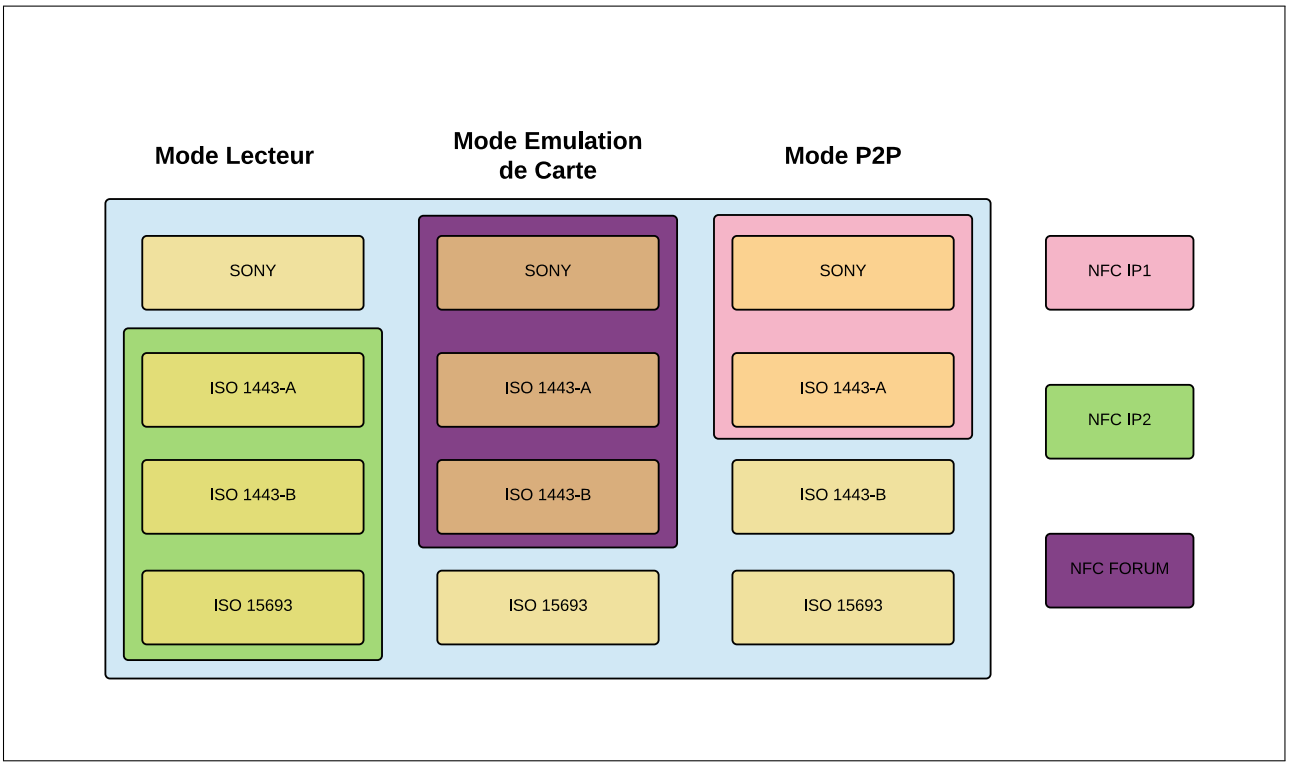

Figure 2.4 Normes de communication NFC

2.2.2 Architecture d'un système NFC

Un système NFC, tel que décrit à la figure 2.5, est composé principalement d'une antenne NFC, d'un contrôleur hôte et d'un contrôleur NFC. Cette architecture peut comporter, selon les besoins, un *Secure Element* (SE) qui constitue un environnement matériel de stockage et d'exécution isolé.

Le contrôleur hôte NFC est le cœur de n'importe quel téléphone mobile. L'interface de contrôleur hôte (HCI) crée un pont entre le contrôleur hôte et le contrôleur NFC, définit

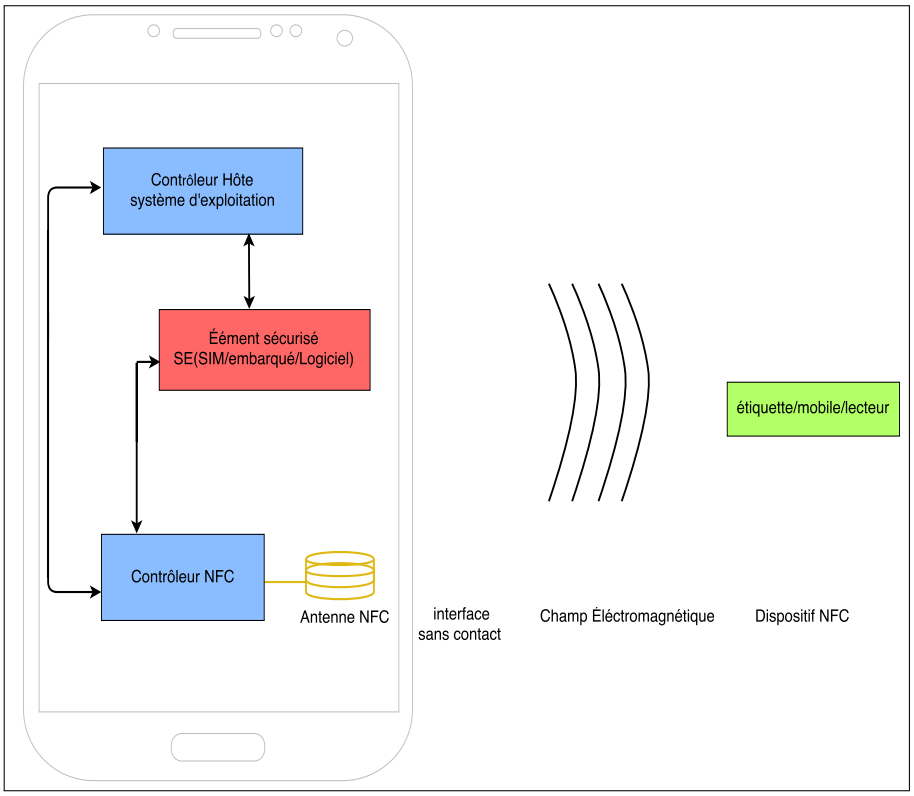

Figure 2.5 Architecture d'un système NFC

le mode de fonctionnement de ce dernier, et traite les données envoyées et reçues. En outre, il maintient l'interface de communication, les périphériques, et l'interface d'utilisateur.

Le contrôleur NFC possède une interface RFID analogue qui comprend un émetteur et un récepteur à la fréquence 13.56 MHz en plus d'un modulateur de charge pour la modulation des réponses en mode émulation de carte. Le contrôleur NFC sert à coordonner plusieurs applications sans contact enregistrées dans l'hôte ou dans différents supports sécurisés, tels que les cartes SIM *Single Wire Protocol* (SWP) ou microSD, ou un élément de sécurité embarqué dans l'appareil. En effet, le protocole SWP définit le standard de communication entre le contrôleur NFC et le SE, ce dernier étant soit une carte SIM, une carte microSD ou un SE embarqué dans le téléphone.

L'antenne NFC est nécessaire afin de permettre de transmettre et capter le signal radio.

2.2.3 Mode de communication

Suivant le type d'alimentation qu'il possède, un dispositif NFC peut fonctionner en deux modes de communication différents.

Mode Actif Dans ce mode de communication, l'initiateur et la cible génèrent leurs propres champs radio pour transmettre des données. Les deux appareils nécessitent donc une source d'énergie indépendante.

Mode Passif Dans ce cas, la cible NFC se contente de l'énergie du champ magnétique généré par l'initiateur pour fonctionner et ne nécessite aucune source d'énergie propre.

2.2.4 Mode d'opération

Comme mentionné précédemment, la différence entre un système NFC et un système RFID réside dans les modes d'opérations et différents types d'usages fonctionnels.

Mode Lecteur de Carte. Le dispositif NFC peut lire et modifier des données d'une cible NFC passive. Ce mode permet de lire des informations en approchant un lecteur NFC, généralement un mobile, devant des étiquettes ou d'autres mobiles fonctionnant en mode d'émulation de carte.

Mode Pair-à-Pair. Spécifié dans la norme ISO18092, ce mode de fonctionnement permet un échange bidirectionnel d'informations entre deux appareils dotés de la technologie NFC. Ce mode d'opération est généralement utilisé pour le transfert de fichiers et le couplage Bluetooth permettant un échange d'information plus rapide.

Mode Émulation de carte. Un dispositif NFC peut également se comporter comme une carte sans contact passive de type ISO 14443 ou FELICA. Dans ce cas, un lecteur externe ne peut pas distinguer entre une carte à puce ou un jeton NFC et un dispositif NFC fonctionnant en mode émulation de carte. Ce mode est notamment utile pour les applications de paiement, de billetterie et de contrôle d'accès.

2.2.5 Protocoles NFC

Sony, *Philips* et *Nokia* sont à l'origine de la mise en place de la technologie NFC, les protocoles de communication NFC nécessitent d'être normalisé afin d'être reconnu par le monde industriel et implémentés par de différents fabricants.

Logical Link Control Protocol (LLCP)

Définit par le NFC Forum, le *Logical Link Control Protocol* (LLCP) est un protocole de couche supérieure d'échange de données entre deux dispositifs NFC qui permet de rendre les communications en P2P plus propres en mettant en place des contrôles de flux. Il est en particulier utile pour reprendre des connections interrompues. En effet, LLCP permet un contrôle de données grâce aux numéros de séquence attribués aux messages permettant de

vérifier qu'ils ont bien tous été reçus. La spécification de ce protocole définit deux types de service :

Service *Connectionless* (sans connexion). Ce mode fournit un service de transmission de données minimal sans accusé de réception

Service *Connection oriented* (orienté connexion). Ce mode fournit un service de transfert de données fiable séquencé et garantit la transmission des unités de données.

Puisque la connexion est établie entre seulement deux dispositifs, LLCP ne fournit pas de multicast/broadcast. De plus, il ne fournit pas la sécurité au niveau de la liaison. LLCP relie les protocoles RFID inférieurs avec des applications et protocoles de haut niveau. Le mode de transport *connection oriented* est responsable de tout échange de données orienté connexion ; commençant lorsqu'une connexion est établie et jusqu'à ce qu'elle s'arrête. Quant au mode de transport *connectionless*, traite tout échange de données sans accusé de réception.

NFC Data Exchange Format (NDEF)

Ce protocole a été spécifié par le NFC-Forum pour définir le format d'échange de données en mode d'opération pair à pair. Le but était de normaliser le format de données utilisées par des objets de NFC Forum notamment les jetons.

Limites NFC sur Android

Le support NFC sur Android a été introduit dans la version 2.3 *Gingerbread*. La première implémentation de la technologie NFC sur Android se limitait au service système NFCService de l'application «NFC system» (package `com.android.nfc`). Cette application englobe les librairies (*drivers*) requises au fonctionnement du contrôleur NFC et implémente les différentes fonctionnalités NFC telles que la découverte d'étiquette (*tag discovery*). Cette première présentation de la technologie NFC au système d'exploitation mobile Android présentait plusieurs limites, notamment l'absence des fonctionnalités du mode émulation de carte. Cette dernière contrainte freinait l'évolution de la technologie NFC sur Android et constituait une entrave majeure au développement de nouveaux services voire même à la migration des services RFID existants sur mobile. Pour contourner ce problème, les fournisseurs de services tel que le paiement mobile et le contrôle d'accès physique utilisaient *Seek for Android*, une implémentation des spécifications *OpenSimAlliance* du système Android qui permettaient l'émulation de carte sur un élément sécurisé matériel, soit une carte SIM ou un élément sécurisé embarqué dans le mobile.

Outre le manque de la fonctionnalité d'émulation de carte, il n'est présentement pas possible d'utiliser le mode d'opération NFC pair à pair en dehors de l'application Beam. Android

Beam est construit sur le protocole *Simple NDEF Exchange Protocol* (SNEP) qui permet un échange transparent de messages NDEF. Malheureusement, le système d'exploitation mobile Android ne dispose pas d'une API publique qui permet d'implémenter directement ce protocole sans passer par l'application Beam, ce qui rend l'expérience client désagréable puisque lors de l'utilisation de Beam, l'utilisateur final est contraint de valider, par un appui sur l'écran, l'envoi de chaque message

2.2.6 Applications de systèmes NFC

Passeport électronique

Les passeports électroniques munis d'une puce NFC, sont décrits par la norme ICAO 9303. Cette puce gère l'accès à différents fichiers contenant les informations nécessaires pour identifier le détenteur du passeport tels qu'une copie des informations de la *Machine Readable Zone* (MRZ), sa photo biométrique, ses empreintes digitales s'il y a lieu ainsi qu'une signature des données du passeport. L'accès aux données du passeport est protégé par trois types de procédures.

- *Basic Access Control* (BAC). Dans ce cas, une clé maître (kseed) est déduite du contenu MRZ. Deux clés sont calculées à partir de kseed et de deux valeurs aléatoires. Elles sont utilisées pour le chiffrement et l'intégrité des données échangées entre le passeport et le lecteur sans contact.
- authentification active (AA). Une clé RSA privée stockée dans le passeport prouve l'authenticité du passeport (mesure anti-clonage).
- *Extended Access Control* (EAC). Une procédure réalisant une authentification mutuelle (Diffie-Hellman sur courbe elliptique) entre le passeport et le lecteur sans contact.

Système de contrôle d'accès

Le contrôle d'accès peut se définir comme un mécanisme de limitation de l'utilisation d'une ressource aux seules entités autorisées. Le contrôle d'accès physique consiste à vérifier si une personne demandant d'accéder à une zone (p.ex. immeuble, bureau, parking, laboratoire, etc.) a les droits nécessaires pour le faire. Le processus de contrôle d'accès commence lorsque l'utilisateur présente ses informations d'identification, typiquement une carte d'employé ou carte d'identité, au lecteur, qui est habituellement monté à côté d'une porte. Le lecteur lit les données de la carte, les traite et les envoie au contrôleur. Le panneau de commande valide le lecteur et accepte les données. Selon la conception globale du système, le contrôleur peut ensuite envoyer les données au serveur ou peut avoir assez d'intelligence locale pour déterminer les droits de l'utilisateur et subordonner l'autorisation finale d'accès. Comme nous avons

mentionné à la section 2.1.1, plusieurs solutions de contrôle d'accès RFID sont disponibles sur le marché, à noter *HID* et *Mifare*. Cependant, nous trouvons très peu de solutions NFC pour le contrôle d'accès. Ceci peut être expliqué par plusieurs raisons, en effet, une solution NFC mobile de contrôle d'accès doit être simple, intuitive, conviviale et unifiée sur les différentes plateformes mobiles : Android, iOS, Blackberry ou autre. De ce fait, l'intégration des services de contrôle d'accès mobile NFC suscite beaucoup d'interrogations. Par exemple, il serait peu convivial si l'utilisateur devait déverrouiller son compte et lancer une application pour ouvrir une porte. De plus, comment est ce que le fournisseur de service gérera le cas ou le téléphone s'éteint ? Généralement, le contrôleur NFC continue à répondre mais avec un UID aléatoire à chaque fois. Jusqu'à récemment, toutes les solutions d'émulation de carte NFC reposaient uniquement sur un modèle basé sur un SE. Ceci nécessite un écosystème sous forme de *Trusted Service Manager* (TSM), qui nécessite à son tour d'intégrations techniques et modèles économiques complexes, expliquant la difficulté de développer des applications de contrôle d'accès sans contact basées sur NFC. Néanmoins, l'arrivée en 2013 du mode *Host Card Emulation* (HCE), avec la version d'Android 4.4 *Kitkat*, a encouragé le fournisseur de solutions de contrôle d'accès physique *HID* à développer une *preuve de concept* d'application de contrôle d'accès mobile NFC mais qui n'est pas commercialisé, à ce que nous savons.

Système de paiement sans contact

La technologie NFC permet d'intégrer les solutions RFID *Paypass/Paywave* aux téléphones mobiles. Les solutions de paiement mobile NFC disponibles sur le marché reposent généralement sur un modèle basé sur un SE sur carte SIM. Ce modèle nécessite une coopération entre les institutions financières, les opérateurs téléphoniques (à l'exception de *Apple Pay*), propriétaires des carte SIM sur lesquelles sont installés les secrets de paiement ainsi que les applets de paiement respectant les spécifications Paypass de Master Card et Paywave de Visa, et une tierce partie de confiance, appelée *Trusted Service Manager* (TSM). Ce TSM est responsable de la gestion du cycle de vie des cartes bancaires mobiles sur la carte SIM. Le rôle du TSM peut être tenu par l'opérateur de télécommunications, dans ce cas le service de paiement sans contact est déployé sur la carte SIM dont il est l'émetteur et dont il a une parfaite maîtrise et connaissance.

Une solution de paiement mobile NFC utilise les standards de paiement reconnus tel que *EMV contactless*, l'équivalent de la spécification EMV aux applications sans contact, et sont par la suite, vulnérables à la même faille discutée à la section 2.1.1. Le seul avantage de ces solutions par rapport à des solutions de paiement RFID est la possibilité de désactiver la puce NFC sur le mobile et de bloquer ainsi les requêtes à la carte mobile quand c'est désiré. D'autres solutions, comme celle de *Samsung Pay*, utilise le mode *Magstripe* ou *bande*

magnétique de la spécification *EMV contactless*. Dans ce cas, la personnalisation de la carte suit les spécifications de personnalisation de carte en bande magnétique, ce qui implique peu, voire pas, de sécurité étant donne que ce mode de fonctionnement consiste à envoyer des données statiques de la carte sans aucune opération cryptographique.

2.2.7 Attaques connues NFC

La technologie NFC cristallise aujourd'hui bien des craintes. En effet, les mêmes menaces dont nous avons discuté dans la technologie RFID sont aussi présentes pour NFC. De plus, le fait que les applications NFC tournent sur un dispositif mobile ajoute d'autres vecteurs d'attaques.

Vecteurs d'attaque mobile. Les téléphones intelligents sont de plus en plus utilisés dans les services bancaires en ligne, la consultation de courriels, l'achat en ligne ou même le contrôle d'accès et le paiement mobile avec la nouvelle technologie sans contact NFC. Cette dépendance croissante n'a pas échappé aux cybercriminels qui exploitent ce nouveau vecteur pour diffuser leurs attaques. Et plus les téléphones intelligents seront utilisés pour effectuer des transactions monétaires, plus ce vecteur de menaces sera utilisé.

Logiciels malveillants *malware*. L'idée de confier des secrets d'un service embarqué dans une carte à puce au mobile est source d'attaque. Un utilisateur malveillant peut vouloir attaquer l'application mobile et son contenu. Un utilisateur ou un programme malveillant peut chercher à attaquer l'application légitime pour en voler des informations ou en tirer profit. Les logiciels malveillants, ciblant plus communément les PC, ont connus ces dernières années une croissance importante sur les appareils mobiles qui attirent de plus en plus les cybercriminels.

Une attaque par logiciel malveillant est la principale menace pour les applications mobiles avec secrets. Indépendamment de la source d'installation —hameçonnage, empoisonnement du magasin d'applications mobiles, magasins d'applications mobiles alternatifs ou sites de téléchargement furtif (*drive-by-download*)— les résultats restent similaires et les attaquants sont susceptibles d'être des mêmes groupes de cybercriminels. Les principales contre-mesures à ce type d'attaque sont : une bonne application des techniques de sécurité disponibles sur le mobile, la sensibilisation des utilisateurs ainsi que les politiques de déploiement des applications sur les magasins d'applications mobiles.

Vol et attaque physique Le vol et les attaques physiques des appareils mobiles sont en recrudescence. La majorité des utilisateurs remarqueront le vol de leur mobile au bout

d'une heure. Dans ce cas, le défi de sécurité est d'empêcher un attaquant ayant un accès physique au mobile d'en extraire des données sensibles avant que la victime rapporte le vol. Certains téléphones proposent des outils de localisation, de verrouillage et de suppression de données à distance.

2.3 Cryptographie sur RFID/NFC

Comme il n'est souvent pas simple de mettre en œuvre des primitives cryptographiques dans les systèmes RFID à faible coût de fabrication, car les étiquettes sont très limitées en termes de puissance de calcul disponible, NFC semble être la bonne solution pour combler les limites de la RFID. En effet, les dispositifs mobiles des dernières générations disposent de capacités de calcul comparables à celles des ordinateurs.

2.3.1 Authentification mutuelle

L'authentification mutuelle par défi-réponse permet à deux entités A et B de prouver leurs identités sans échanger de secrets au moment de l'authentification. En effet, après établissement de lien, l'entité initiant la connexion, A, reçoit un défi sous la forme d'un nonce, une valeur idéalement imprévisible et à usage unique, de la part de la deuxième entité B. A prouve son identité en répondant à ce défi en utilisant un secret partagé k, une clé cryptographique, nécessaire pour calculer la réponse et pour la vérifier. De cette manière, un attaquant n'est pas capable de répondre à la place de A même s'il a écouté tous les messages échangés entre A et B.

2.3.2 Authentification mutuelle à cryptographie symétrique

La cryptographie symétrique repose sur le concept de clé secrète partagée. Cette même clé est utilisée et pour le chiffrement et pour le déchiffrement de données. D'où la nécessité de sécuriser son transfert depuis l'entité de gestion de clés ainsi que son stockage. L'avantage d'un système à clé symétrique est qu'il est beaucoup plus rapide qu'un système à clé publique, de sorte que de grandes quantités de données peuvent être chiffrées de manière efficace. Cependant, la gestion des clés devient un fardeau lorsque le nombre d'utilisateurs augmente. Comme expliqué dans la section 2.1.1, les étiquettes RFID de la 2ème génération sont capables de faire de l'authentification mutuelle à clé symétrique en utilisant un mécanisme de défi/réponse. La figure 2.6 illustre ce mécanisme.

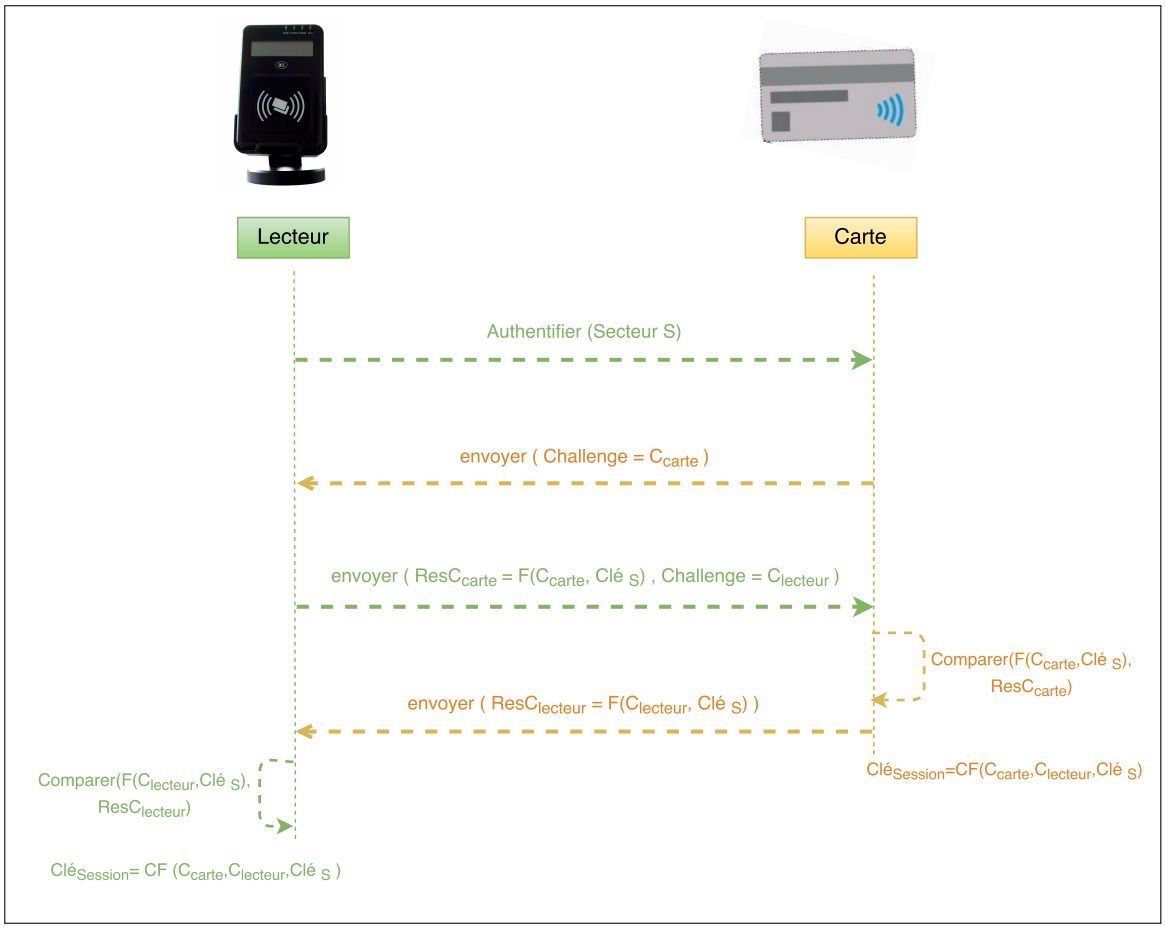

Figure 2.6 Authentification mutuelle des cartes de 2ème génération

2.3.3 Authentification mutuelle à cryptographie asymétrique

Dans la cryptographie asymétrique, une paire de clés publique/privée est générée. Les clés publiques sont publiées et accessibles à tout le monde, alors que les clés privées sont gardées secrètes. Dans un algorithme de chiffrement, les clés publiques sont utilisées pour chiffrer le message à envoyer à une deuxième entité où le déchiffrement du messages se fait en utilisant la clé privée associée. Un autre cas utile est d'utiliser une clé privée pour signer un message qui sera vérifié avec une clé publique. Les clés sont générées en utilisant une fonction à sens unique, de sorte qu'il est mathématiquement impossible de deviner une clé privée à partir de la clé publique dans un temps polynomial. Alors que les clés publiques fournissent des systèmes sécurisés forts, l'inconvénient est que les opérations cryptographiques utilisant des clés de grande taille sont coûteuses en calculs, chose qui n'est désirable pour les appareils avec une puissance de calcul limitée.

Notons comme exemple d'application RFID utilisant cette méthode, les cartes bancaires

sans contact. En effet, l'authentification de données de carte en mode DDA implique l'utilisation d'une paire de clés publique/privée.

2.4 Cryptographie à courbes elliptiques (ECC)

La cryptographie à courbe elliptique (ECC) est une primitive de cryptosystème à clé asymétrique définie à partir d'un groupe défini sur des courbes elliptiques sur un corps fini $\mathbb{Z}_p^*$. Comme dans tous les systèmes de cryptographie à clé publique, aucun secret n'est partagé entre l'émetteur et le récepteur, le déchiffrement est contrôlé par une clé privée gardée secrète, le chiffrement quant à lui est fait par une clé connue publiquement.

2.4.1 Généralités

On appelle une courbe elliptique sur un corps $\mathbb{K}$ toute courbe d'équation :

$$Y^2 = X^3 + aX + b \tag{2.1}$$

tel que a et b sont deux éléments de $\mathbb{K}$ tels que $\Delta = 4a^3 + 27b^3 \neq 0$

Soit une courbe elliptique E, on définit un point caractéristique à l'infini de E qu'on appelle O. On définit sur les points de E une addition notée + qui fera de l'ensemble des points sur E, un groupe abélien $\mathbb{E}$ de point neutre O. Soit P et Q deux points distincts de la courbe E, et R un troisième point ou la droite qui passe par P et Q recoupe la courbe. Le symétrique $-R$ de ce point par rapport à l'axe des abscisses constitue l'addition de P et Q, c.à.d. $P + Q = -R$. Cette loi d'addition satisfait les propriétés suivantes :

- Fermeture : $P + Q = -\mathrm{R} \in \mathbb{E}$.
- Commutativité : $P + Q = Q + P$.
- Identité : $P + O = P + O = P$.
- Associativité : $(P + Q) + R = P + (Q + R)$

2.4.2 Paramètres du domaine

Les entités en communication doivent se mettre d'accord sur tous les éléments définissant la courbe elliptique, ce qu'on appelle les paramètres du domaine.

- a, b : deux éléments qui définissent l'équation de la courbe elliptique
- G : point générateur de la courbe elliptique E
- h : l'ordre de la courbe elliptique

2.4.3 Génération de clés

Soit E une courbe elliptique définie par l'équation $Y^2 = X^3 + aX + b$, p un très grand nombre premier pour former le corps $\mathbb{Z}_p$ et G le point générateur de E.

1. on choisit aléatoirement un entier $s \in [1, p-1]$.
2. on calcule $Q = s * G$ sur le groupe de courbe elliptique $\mathbb{E}(\mathbb{Z}_p)$.

Le couple (Q, s) constitue une paire de clé publique et privée.

La ECC permet notamment de réduire la taille des clés et données échangées comparant à d'autres cryptosystèmes tels que RSA, *Digital Signature Algorithm* (DSA) ou Diffie-Hellman tout en gardant un niveau de sécurité équivalent.

2.4.4 *Elliptic Curve Diffie-Hellman* (ECDH)

Il s'agit d'un algorithme d'échange de clés Diffie et Hellman sur une courbe elliptique. Comme son homologue classique, ECDH peut être utilisé pour créer une clé secrète partagée entre deux entités communicant sur un canal non sécurisé.

Nous décrivons dans ce qui suit les étapes d'échange de clés entre Alice et Bob .

1. Alice et Bob se mettent d'accord sur une courbe elliptique E de paramètres de domaine (p, a, b, G, n, h).
2. Alice choisit un nombre aléatoire K_A envoie à Bob $K_A * G$
3. Bob choisit un nombre aléatoire K_B envoie à Alice $K_B * G$.
   (L'échange de $K_B * G$ et de $K_A * G$ n'a pas besoin d'être sécurisé)
4. Alice calcule $K_A * (K_B * G)$
5. Bob calcule $K_B * (K_A * G)$
6. Les propriétés de commutativité et d'association du groupe de E permettent de conclure qu'Alice et Bob disposent de la même clé secrète, soit :

$$K_A * (K_B * G) = (K_A * K_B) * G = (K_B * K_A) * G = K_B * (K_A * G)$$

2.4.5 *Elliptic Curve Digital Signature Algorithm* (ECDSA)

Le ECDSA est l'analogue de l'algorithme DSA sur les courbes elliptiques. DSA a été spécifié dans la norme de signature numérique *Digital Signature Standard* (DSS) du *Federal Information Processing Standard* (FIPS). La sécurité de cet algorithme repose sur le problème de logarithme discret dans les sous groupes d'ordre premier de $\mathbb{Z}_p^*$. Nous décrivons dans ce qui suit les étapes de signature et de vérification de signature de ECDSA.

Signature. Soit une entité A d'une courbe elliptique E de domaine $D = (p, a, b, G, n, h)$ et (Q, d) le couple de clés publique et privée associé à A et générés comme expliqué dans la section 2.4.3. La phase de signature prend en paramètre d'entrée un message m à signer et a en sortie une signature à ce message m.

1. Choisir un nombre aléatoire $k, k \in [1, n-1]$.
2. Calculer $k * G = (x_1, y_1)$.
3. Calculer $r = x_1 \mod n$. Si $r = 0$ revenir à l'étape 1.
4. Calculer $k^{-1} \mod n$.
5. Calculer SHA-1(m) et le convertir en entier e.
6. Calculer $s = k^{-1} * (e + d * r) \mod n$. Si $s = 0$ revenir à l'étape 1.
7. (r, s) constitue la signature du message m par l'entité A.

Vérification de signature. Pour que B vérifie la signature (r, s) du message m par l'entité A dont (Q, d) est le couple de clés publique et privée, B doit recevoir en premier lieu les paramètres du domaine de A, soit $D = (p, a, b, G, n, h)$, et la clé de vérification de la signature, soit la clé publique Q.

1. Vérifier que $r, s \in [1, n-1]$.
2. Calculer SHA-1(m) et le convertir en entier e.
3. Calculer $w = s^{-1} \mod n$.
4. Calculer $u_1 = e * w \mod n$ et $u_2 = r * w \mod n$.
5. Calculer $X = u_1 * G + u_2 * Q$.
6. Si $X = O$ (le point neutre de $\mathbb{E}$), rejeter la signature sinon calculer $v = x_1 \mod n$ tel que x_1 est la composante en x de X.
7. Accepter la signature uniquement si $v = r$.

2.5 Conclusion

Nous avons vu dans ce chapitre un aperçu des menaces auxquelles les systèmes NFC sont exposés. L'utilisation de cette technologie sur les téléphones intelligents donne l'espoir d'une solution plus sécuritaire. Cependant, il ne faut pas négliger les nouvelles menaces introduites avec un tel système notamment les logiciels malveillants. La mise en place d'un canal sécurisé est la meilleure approche pour sécuriser une communication NFC contre les attaques citées. Une telle approche nécessite à la fois l'implémentation d'un protocole d'authentification permettant d'établir ce canal sécurisé ainsi que garantir un stockage sécurisé des clés

cryptographiques utilisées dans ce protocole. Nous discutons dans les chapitres suivants des différentes solutions de stockage de clés cryptographiques sur le système Android et nous proposons un protocole d'authentification mutuelle permettant d'établir un canal sécurisé entre deux dispositifs NFC communicant ensembles.

CHAPITRE 3

ANALYSE DE SOLUTION DE STOCKAGE DE CLÉS CRYPTOGRAPHIQUES

Comme discuté au chapitre 2, l'utilisation de la RFID à des fins d'authentification est source de menace. Comme les systèmes RFID traditionnels manquent de ressources nécessaires aux calculs cryptographiques complexes, les plateformes mobiles peuvent améliorer nettement cette faiblesse avec leur capacité de calcul comparable à celle des ordinateurs personnels. De ce fait, la technologie NFC rendue disponible sur des plateformes mobiles semble plus prometteuse en termes de performances. Cependant, si les plateformes mobiles sont capables d'effectuer des calculs cryptographiques à clé publique en un laps de temps raisonnablement court, ce mariage NFC/mobile n'échappe pas aux cybercriminels qui peuvent exploiter ce nouveau vecteur de menace pour diffuser leurs attaques. Ainsi, sécuriser le stockage des secrets cryptographiques (clés) utilisés dans des protocoles cryptographiques à clés publique dans ce contexte est indispensable pour la sécurité d'une solution d'authentification NFC mobile à clé publique. En effet, l'utilisation d'ICP dans un contexte NFC requiert que le détenteur de carte prouve qu'il dispose d'une clé d'authentification à clé publique en effectuant des opérations cryptographiques que la deuxième entité, le lecteur, peut vérifier. Nous nous intéressons dans notre étude au système d'exploitation mobile *Android* comme la plateforme mobile dotée de la technologie NFC ayant la plus grande part du marché. Nous commençons par présenter d'abord la plateforme Android ainsi que ses différents mécanismes de sécurité. Ensuite nous nous intéressons aux différentes solutions de stockage de secret sur Android ainsi que les environnements définis dans un contexte NFC notamment les notions du SE et de *Trusted Execution Environment* (TEE). Nous concluons le chapitre avec une comparaison entre ces différentes solutions en expliquant les limites et les avantages selon le champs d'application.

3.1 Mécanismes de sécurité sur Android

Android est un système d'exploitation mobile *open source*, basé sur un noyau Linux comme illustré la figure 3.1 et actuellement développé par Google. Les applications Android sont écrites en Java et transformées en un format légèrement différent connu comme « .dex ». Elles sont par la suite exécutées dans la machine virtuelle Dalvik (MVD) qui fournit une couche d'abstraction avec le matériel. Android dispose de différents mécanismes de sécurité

que nous détaillons dans ce qui suit.

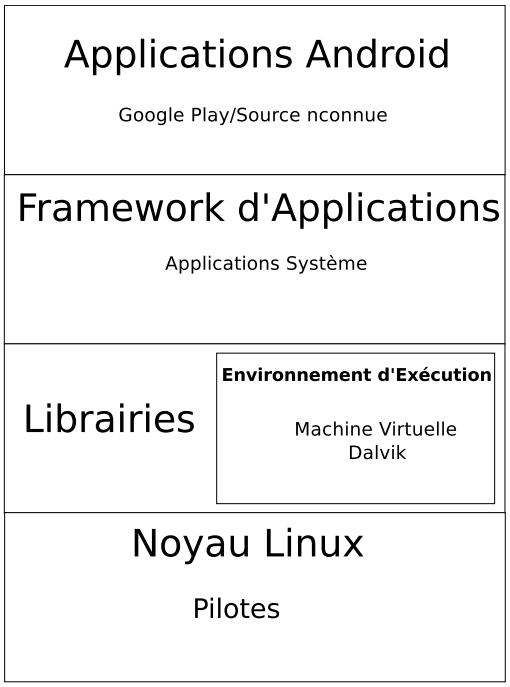

Figure 3.1 Architecture du Système Android

3.1.1 Sécurité du noyau Linux

Au niveau du système d'exploitation, la plateforme Android assure la sécurité du noyau Linux ainsi qu'une communication inter-processus sécurisée pour garantir un échange sécurisé entre applications tournant sur différents processus. Ces mesures de sécurité au niveau du système d'exploitation garantissent que même le code écrit en langage natif soit « sandboxé », c'est-à-dire que chaque application est exécutée dans une bulle isolée du reste du système. De ce fait, une application potentiellement dangereuse peut être lancée sans risque d'infecter le système. De plus, le noyau Linux apporte plusieurs autres mécanismes de sécurité au système d'exploitation mobile Android y compris :

Isolation des processus des applications. La plateforme Android profite du mécanisme de *Sandboxing* du noyau Linux pour isoler les applications les unes des autres ainsi que des ressources systèmes. En effet, chaque application Android possède un identifiant unique «UID» qui lui est assigné au moment de l'installation permettant au système d'exploitation une isolation entre les applications au niveau de la mémoire.

Gestion d'accès aux fichiers de l'application. Dans un environnement UNIX, le modèle de permission sur le système de fichier garantit qu'un utilisateur ne peut pas modifier ou lire les fichiers d'un autre utilisateur. Dans le cas du système Android, chaque

application est propriétaire des ressources qui lui sont assignées, ainsi ces ressources ne peuvent pas être lues ou modifiées par une autre application que si leur propriétaire le permet explicitement.

3.1.2 Sécurité au niveau de l'application

Les applications Android sont souvent écrites en langage de programmation Java mais peuvent aussi intégrer du code natif écrit en *C/C++* à travers le *Java native interface* (JNI) et le kit de développement officiel *Native Development Kit* (NDK). Ces applications sont empaquetées dans un seul fichier appelé « APK » incluant le bytecode de l'application (`classes.dex`), le manifeste de l'application, ses ressources et les signatures numériques, et tournent sur la MVD. Ce fichier APK n'est qu'une simple archive qui contient différents éléments importants pour juger la sécurité de l'application dont le `AndroidManifest.xml` et le certificat de l'application. En effet, le manifeste fournit les informations concernant l'application dont la liste de permissions que l'application sollicite du système.

Modèle de permission. Android utilise un modèle de permissions à granularité fine concentré sur l'utilisateur, qui exige à l'application de prédéfinir les permissions dans le fichier Manifest. En effet, au moment de l'installation de l'application sur le système, l'utilisateur doit accepter une liste de permissions définies.

Signature du paquet de l'application. Avant d'être publiées sur le magasin d'applications *Google Play*, les applications doivent être signées. Cette obligation de signature est notamment nécessaire pour identifier l'auteur de l'application, assurer l'intégrité du package APK, autoriser certaines interactions entre des applications signées par la même clé, et garantir la sécurité du système lors de la mise à jour de l'application en acceptant uniquement des mises à jours ayant la même signature que celle de l'application publiée initialement sur le magasin d'applications mobiles. Le certificat peut être généré en utilisant *keytool* ou *jarsigner* manuellement ou automatiquement (Pocatilu, 2011).

L'ensemble des applications signées par le même certificat s'exécutent sous la même identité de groupe Unix ou *gid*. Ce mécanisme d'identité de groupe constitue un « carré de sable » (*sandbox*) protégeant les applications du même groupe des autres applications tournant sur le même système, quoiqu'il peut être aussi source d'attaques. En effet, ces applications peuvent aussi partager le même gid ce qui peut conduire à l'apparition d'applications malveillantes k-aire. De ce fait, chaque application ne possède pas assez de permissions individuellement pour éveiller le soupçon du système, mais la combinaison de ces applications dans une même

identité de groupe forme une application malveillante avec des permissions dangereuses permettant de compromettre la confidentialité et l'intégrité des données du téléphone.

3.1.3 *Security enhancement for Android*

Security enhancement for Android (SE for Android) est un projet visant à identifier et combler les faiblesses de sécurité du système Android en implémentant un mécanisme similaire à celui de *Security enhancement for Linux* (SE Linux), qui vise à améliorer la séparation entre applications tournant sur le même système. Sur les appareils Android *rootés*, les utilisateurs malveillants peuvent installer des applications qui lisent les mots de passe, qui envoient des messages indésirables aux clients, qui téléchargent des documents confidentiels sur Internet ou qui activent en toute discrétion des ressources telles que l'appareil photo ou le microphone. Similairement à SE Linux, SE for Android définit les utilisateurs ou les applications pouvant accéder à des ressources et à des fichiers donnés. Il applique le *Mandatory Access Control* (MAC) avec des fichiers de stratégie. De ce fait, SE for Android peut assurer la sécurité du système d'exploitation en créant différents domaines de sécurité. Dans chaque domaine, les applications bénéficient de l'autorisation minimale permettant leur strict fonctionnement. Ce processus limite les dommages à une seule zone en cas d'infection, les autres zones n'étant pas compromises. SE for Android peut fonctionner en trois modes différents :

Mode désactivé. SE for Android n'est pas activé et aucun fichier de stratégie n'est chargé.

Mode permissif. Dans cet état, le fichier de stratégie SE for Android est chargé mais n'est pas appliqué. L'accès à des ressources non autorisées n'est donc pas bloqué mais tracé dans des fichiers journaux à des fins de test et débogage.

Mode applicatif. SE for Android charge et applique la stratégie de sécurité. Dans ce cas, toute application malveillante qui tente d'accéder à une ressource non autorisée est bloquée.

3.2 Stockage de données sur Android

En plus de la sécurité des échanges d'informations, une solution d'authentification NFC sur mobile requiert aussi un stockage sécurisé des clé cryptographiques utilisées dans ce processus. En effet, les données sensibles comme les clés cryptographiques ou les données bancaires doivent être stockées dans une mémoire inviolable, plus sécurisée et plus résistante que le téléphone lui-même. Android dispose de plusieurs options pour stocker les données persistantes de ses applications. Ainsi, le choix entre ces options dépend des spécifications de l'application, de type, de volume de données à stocker et de la technologie utilisée (solution SE vs. HCE) dans le cas d'un service NFC. La perte de l'appareil mobile est un problème

très commun qui doit être pris en considération dans la conception et l'implémentation de services mobiles. Un attaquant ayant un accès physique à l'appareil peut effectuer différents types d'attaques allant de voler des données personnelles au vol d'informations sensibles telles que des données bancaires, jetons d'authentification ou aussi clés cryptographiques. Ce scénario pourrait être encore pire si le dispositif est rooté.

3.2.1 Stockage logiciel

Préférences d'application Shared Preferences

Les préférences d'application permettent de stocker d'une façon persistante des données primitives sous la forme de couples de clé/valeur. Suivant le mode de définitions de préférences, plusieurs applications peuvent y accéder.

Stockage Interne

Android permet de stocker les fichiers des applications dans la mémoire interne du mobile. La mémoire interne est séparée en deux zones : une zone réservée pour les applications «système» et leurs données non accessibles aux utilisateurs, et une zone pour installer les applications «non système» ainsi que les données des utilisateurs telles que les photos, les vidéos et autres fichiers.

Stockage Externe

Le système d'exploitation mobile Android permet aussi de monter un espace de stockage externe tels qu'une carte SD où les applications peuvent stocker leurs fichiers de volumes importants. La majorité de ces dispositifs de stockage externe disposent d'un système de fichier VFAT où le mécanisme de contrôle d'accès aux fichiers de Linux ne peut pas fonctionner. Dans ce cas, un chiffrement de données sensibles est nécessaire. De plus, à partir de l'API8, Android propose une option d'enregistrer le répertoire d'application «.APK» dans le dispositif de stockage externe. Cette option permet notamment de libérer de l'espace mémoire interne du mobile.

3.2.2 Stockage matériel

Dans ce type de stockage l'hôte mobile ne devra pas stocker de données sensibles. Le mobile héberge une application Android appelée aussi interface homme machine (HMI) responsable de la gestion de la communication avec l'utilisateur. Elle échange avec une autre application

hébergée sur environnement sécurisé matériel pour effectuer des opérations cryptographique, stocker des données sensibles ou communiquer avec le contrôleur NFC.

Élément de Sécurité

La notion du SE a été développée par le *NFC Forum* comme solution de stockage isolé. Il s'agit d'un élément de sécurité matérielle dans lequel sont hébergées des applications et des données dites « sensibles ». Les dernières générations sont équipées de co-processeurs cryptographiques capables d'implémenter les algorithmes standards comme DES, AES et RSA. La sécurité du SE repose sur le fait que ce dernier est capable de communiquer directement avec la borne sans contact externe, ce qui empêche la visibilité de la transaction à l'OS du téléphone, souvent peu ou pas suffisamment protégé. Cette solution de stockage, bien que recommandée par le *NFC Forum* est complexe et nécessite la collaboration de plusieurs intervenants comme le TSM et le fournisseur du SE comme illustré dans la figure 3.2. En effet, le fournisseur de service sera incapable de déployer sa solution sur le SE et aura besoin de permission et certification du fournisseur du SE. Dans cette architecture, le TSM, responsable du déploiement du service et des données sensibles sur le SE, joue le rôle intermédiaire entre le fournisseur du matériel et le fournisseur de service.

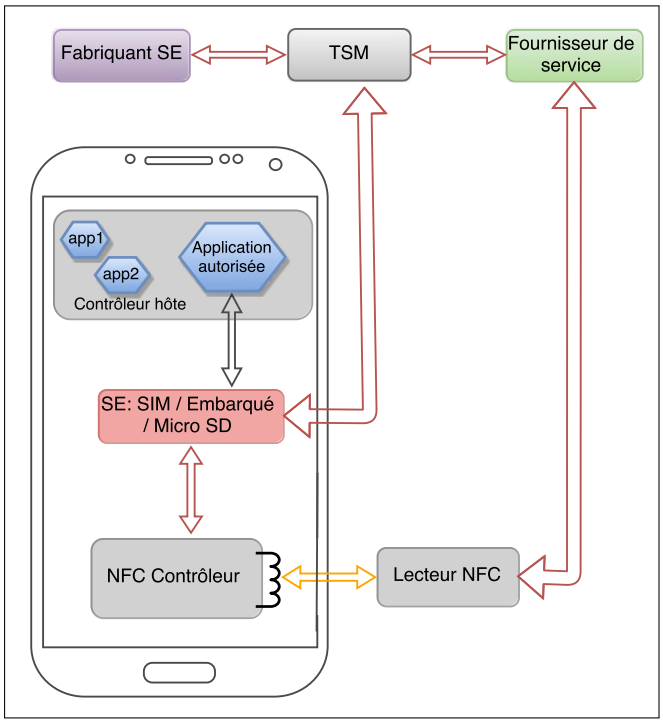

Figure 3.2 Architecture d'une solution Secure Element (SE)

Les SE, tels que définis par *NFC Forum* se divisent en trois familles : les SE embarqués

sur le mobile, les SE dans la carte *Subscriber Identity Module* (SIM) et les SE sous forme de Micro-SD. Nous n'allons pas détailler ce dernier format de SE, puisqu'à l'heure actuelle, il n'existe pas de solutions disponibles sur le marché de ce type de SE ainsi que la minorité des modèles de téléphone compatibles (ayant NFC et disposant d'une Micro-SD).

SE embarqué. Android propose à partir de sa version 2.3.4 une API non publique pour manipuler le SE intégré aux téléphones ou tablettes récents. Malheureusement, cette API nécessite un privilège système pour pouvoir déployer des application sur ses SE embarqués, qui ne peut être accordé qu'au fournisseur du téléphone ou à Google pour la famille Nexus. En effet, pour pouvoir y déployer des applications ou manipuler les données des SE embarqués, l'application doit être signée par le signataire de la plate-forme. Android 4.0.4 (API niveau 15) modifie cela en utilisant à la place une liste blanche de signatures autorisées (/etc/nfcee_access.xml). Cette liste est figée sur la plate-forme mais peut être mise à jour *Over The Air* (OTA) par le constructeur. Ce fichier indique la signature et la liste des packages valides.

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
  <signer android:signature="30820...90">
    <package android:name="org.foo.nfc.app">
    </package>
  </signer>
</resources>
```

Avec un accès administrateur, il est possible de remonter la partition en mode lecture/écriture pour modifier ce fichier. Seule la permission NFC est alors nécessaire. L'application doit également ajouter une librairie.

```
<uses-library
   android:name="com.android.nfc\_extras"
   android:required="true">
```

Ensuite, une API minimaliste permet de communiquer des trames binaires avec le SE.

```
NfcAdapterExtras adapterExtras =
  NfcAdapterExtras.get(NfcAdapter.getDefaultAdapter(context));
NfcExecutionEnvironment nfceEe =
  adapterExtras.getEmbeddedExecutionEnvironment();
nfcEe.open();
byte[] response = nfcEe.transceive(command);
nfcEe.close();
```

Google utilise cela pour Google Wallet. Il s'agit du portefeuille développé par la compagnie Google initialement déployé sur leurs appareils Nexus en 2011, pour y placer toutes les cartes de crédit, données particulièrement sensibles. Le fonctionnement de ce portefeuille dépend de deux applications installées sur le SE : l'applet de contrôle du portefeuille et l'applet de simulation de carte bancaire, compatible avec les applications *Paywave/Paypass*. La première applet active ou désactive l'applet de simulation de carte suivant la saisie d'un code PIN par l'utilisateur. Cette dernière, une fois activée, simule les spécifications EMV classiques comme les cartes bancaires. De ce fait, un mobile Android ayant cette application et présent devant un terminal de paiement peut fonctionner comme une carte bancaire classique et NFC sur Android est paramétré pour faire transiter les trames vers le SE correspondant.

Ce composant peut être très utile pour pouvoir sauver les clés cryptographiques utilisées dans nos algorithmes d'authentification NFC puisque seul le TSM du composant est habilité à installer des applications dans le SE.

SE sur une carte SIM. Ce type de SE constitue aujourd'hui l'approche la plus répandue de système d'émulation de carte NFC. Dans ce cas, Android est découpé en deux parties distinctes : un processeur est chargé du système d'exploitation et un autre processeur économe chargé des communications radios. Comme illustré dans la figure 3.3, le système d'exploitation est contraint d'utiliser le SWP pour que la carte SIM et le contrôleur NFC puissent communiquent. Une implémentation de ce protocole est disponible dans Open Mobile API de SimAlliance. Bien que certains constructeurs de téléphones mobiles tels que *Samsung* et *Sony* commencent à l'intégrer dans leurs nouveaux appareils mis sur le marché, cette API n'est malheureusement pas disponible pour plusieurs autres modèles de téléphones. Pour cela, les opérateurs qui offrent ce service fournissent des images personnalisées du système d'exploitation Android incluant cette API.

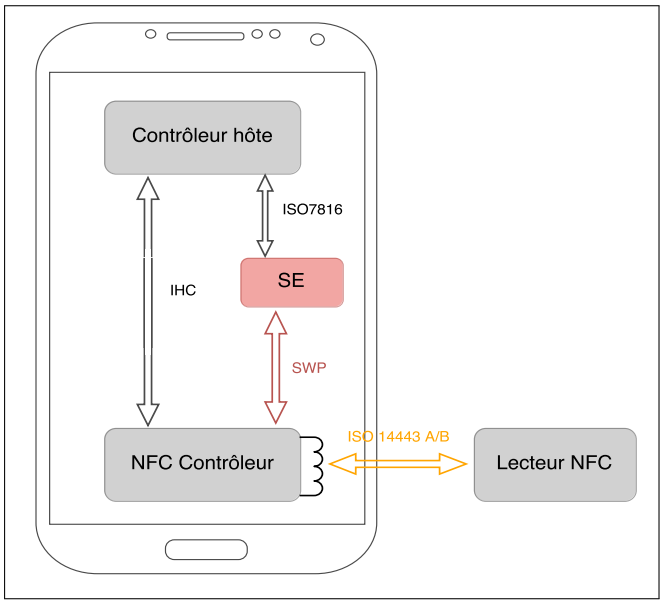

Figure 3.3 Architecture d'une solution Secure Element sur une carte SIM

Trusted Execution Environment (TEE)

Global Platform spécifie le fonctionnement logiciel du SE, et écrit les spécifications pour permettre l'interopérabilité et réduire les délais de mise sur le marché. *Global Platform* publie non seulement des spécifications « Card » pour l'élément sécurisé, mais également des spécifications « Device » pour les mobiles et « System » pour les plateformes mobiles. Dans ce contexte, cet organisme a défini un environnement d'isolation pour les *System on Chip* (SoC) afin de séparer le traitement de code et de données sensibles de l'environnement d'exécution et la mémoire principales. Cet environnement est dédié à la gestion des fonctions sensibles et du code dont l'exécution dans l'OS normal exposerait la sécurité aux attaques. Les fonctionnalités sensibles sont exportées dans le *Trusted Execution Environment* (TEE) dans des applications de confiance (*trustlets*), mais leur gestion demeure extérieure au TEE. *Global Platform* définit deux principales propriétés du TEE comme environnement de sécurité isolé, à savoir la sécurité et l'isolation.

Fonctions de sécurité

Stockage sécurisé. Les opérations cryptographiques traitant des données sensibles telles que des clés cryptographiques sont exécutées à partir des trustlets installées sur le TEE. De ce fait, les clés cryptographiques ne quittent jamais le TEE.

HMI sécurisée. Cette fonction est nécessaire pour garantir que les informations affichées à l'utilisateur, et entrées par ce dernier sont sécurisées et ne peuvent pas être interceptées

par une autre application malveillante.

Canaux sécurisés avec le SE. Une option d'utiliser le TEE est d'héberger l'application mobile HMI responsable de la gestion de communications entre l'utilisateur et le SE. Dans ce cas, le fournisseur de service ne délègue aucune responsabilité au hôte mobile et tout le traitement se fait entre le TEE et le SE. Dans ce cas *Global Platform* définit le *Secure Channel Protocol* pour sécuriser les interactions entres ces deux composants de sécurité.

Fonctions d'isolation

Stockage sécurisé. Le TEE garantit un stockage de données sensibles dans un espace mémoire isolé de celui de l'OS Android.

Isolation entre les différentes applications du TEE. Cette fonctionnalité permet la coexistence de différentes applications de différents fournisseurs de services sur un même TEE en garantissant une isolation des données et du code de ces applications.

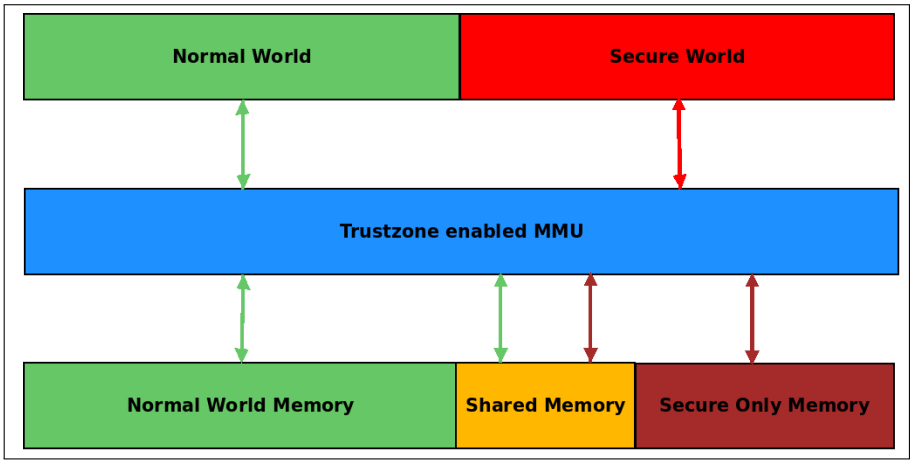

Figure 3.4 Gestion de la mémoire dans Trustzone TEE

Stockage Matériel : Android Key Store

Le *Android Key Store* est un système de gestion de clés cryptographiques sur Android, qui introduit un API publique pour le stockage et l'utilisation des clés privées des applications. En effet, le système Android lance un service système codé en C, permettant de protéger les différents secrets/certificats numériques. Les applications peuvent communiquer avec ce dernier, via un socket local, pour demander le déblocage du conteneur, y stocker des secrets ou de les retrouver. Ces secrets sont isolés par application. Le système Android impose l'utilisation d'un verrou d'écran qui servira pour chiffrer le conteneur de secret à l'aide d'un dérivé

de ce verrou, appliqué suffisamment de fois pour résister à une attaque de force brute. En effet, si le téléphone est verrouillé, le conteneur est bloqué et lorsque l'utilisateur déverrouille le téléphone, le conteneur est ouvert aux applications.

Un stockage matériel est rendu disponible dans l'API 18 Android 4.3 permettant une sécurité supplémentaire des clés privées des application afin de rendre impossible l'extraction de données. Nexus 4 introduit une application de confiance «Keymaster» qui tourne sur *Qualcom Secure Execution Environment* (QSEE), environnement sécurisé basé sur «trusted zone» de ARM. Cette application, se contente d'envoyer des commandes à l'API QSEE et analyse les réponses sans effectuer les opérations cryptographiques.

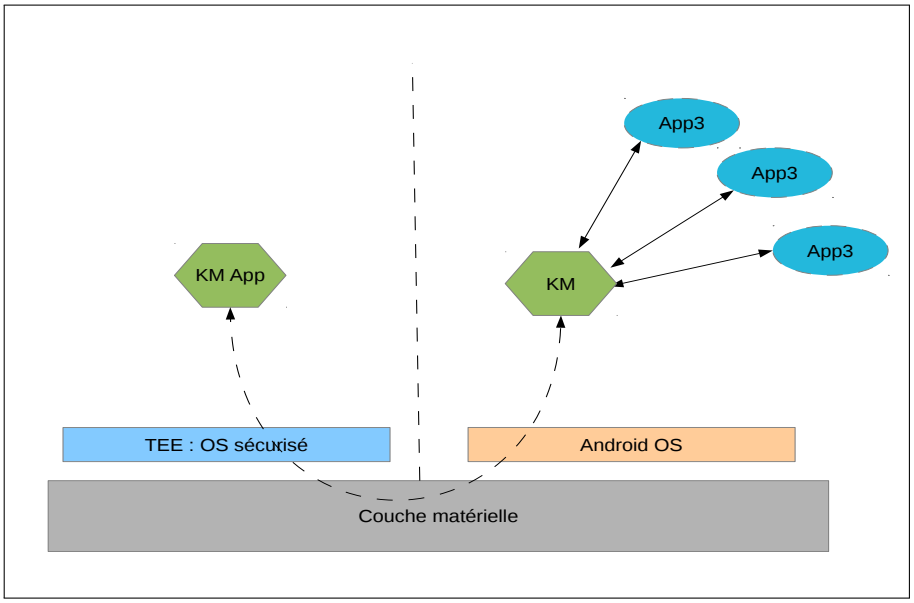

Figure 3.5 Environnement d'exécution fiabilisé (TEE)

Bien que cette solution, semble être avantageuse puisqu'elle permet de sécuriser les clés cryptographiques des applications sans avoir besoin d'installer une application sur l'élément de sécurité SE ou TEE et par la suite contourner les exigences de collaborer avec les fournisseurs des SE ou même les TSM, elle reste limitée par le fait que cet élément matériel QSEE hébergeant l'application *Keymaster* n'est pas disponible sur tous les modèles de téléphone et nous serons, encore une fois, limités par la disponibilité et la compatibilité matérielle.

3.3 Comparaison d'une solution d'émulation de carte logicielle et matérielle

Les solutions disponibles de stockages de clés cryptographiques des services d'authentification NFC sur mobile comme nous avons présenté sont principalement divisées en deux

familles : des solutions purement logicielles ou des solutions purement matérielles. Les premières applications d'émulation de carte NFC sur Android, notamment le paiement mobile et le transport en commun, utilisaient principalement des solutions SE. En effet, ce choix a été poussé par une limite importante de la technologie NFC sur Android, soit l'absence d'implémentation logicielle du mode d'émulation de carte HCE. Cette dernière limite a poussé l'adoption d'une approche SE matériel. Cependant, le mode HCE a été introduite dans Android à partir de sa version 4.4 *Kitkat*. Cette nouvelle fonctionnalité a ouvert plus d'horizons aux fournisseurs de service avec un modèle d'affaires plus simple et une solution plus facile à maintenir. En effet, une solution HCE nécessite moins d'intervenants comme le TSM ou le fournisseur du SE. Cependant, une telle solution demande plus de réflexion en ce qui concernent le stockage de clés cryptographiques.

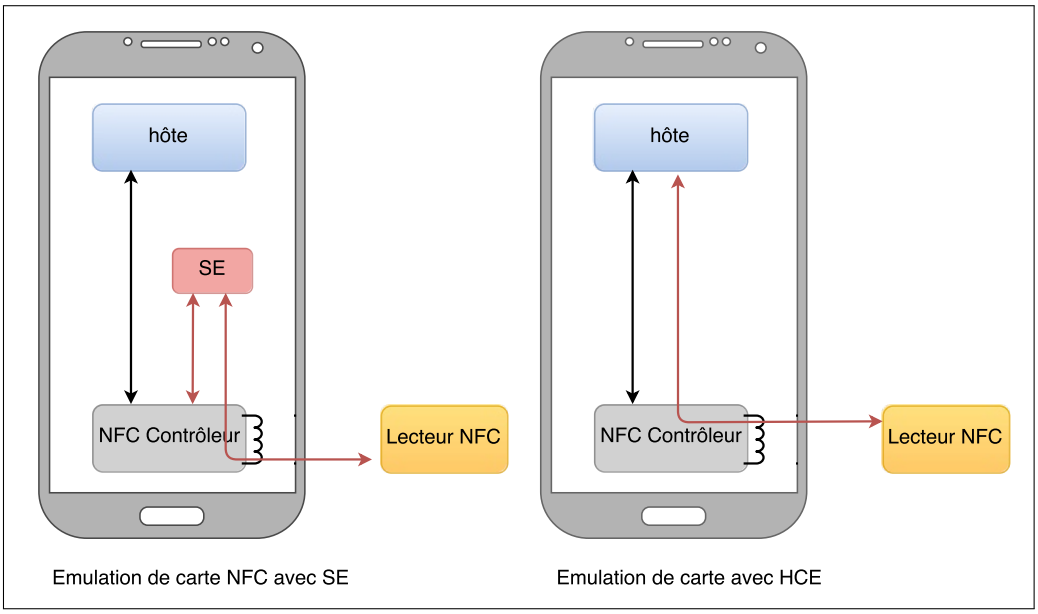

Figure 3.6 Host Card Emulation (HCE) vs. Secure Element (SE)

Avec la levée du verrou d'émulation de carte NFC sur Android, plusieurs fournisseurs de service notamment des institutions financières se sont intéressés à fournir une solution de paiement HCE. Notons par exemple, la Banque Royale du Canada (RBC) qui a présenté la première solution de paiement HCE au Canada en septembre 2015 et Google qui a aussi déployé sa propre solution HCE *Android Pay* aux États-Unis lors du dernier trimestre de 2015. Ces deux solutions reposent sur un nouveau concept : la *Tokenisation* qui consiste à substituer les données de la carte par des *token* (jetons) jetables limitées soit par une durée de vie ou un nombre d'utilisation fixé. Le but de la tokenisation dans ce contexte est de réduire l'impact de récupération de données sensibles par un logiciel malveillant exécuté dans

la mémoire du téléphone. En effet, en cas de compromission de ces données, le risque reste limité puisque les jetons compromis sont limités dans le temps et vont expirer après un certain nombre d'utilisation. Ce qui implique le stockage des éléments sensibles (données bancaires, clés cryptographique) dans un serveur distant, généralement hébergé pas le fournisseur de service lui même. De plus, une telle solution fonctionne uniquement en mode connecté, pour effectuer le remplacement des jetons expirés contrairement à une solution SE qui n'a besoin de connectivité qu'au moment de provisionnement de la carte sur le mobile. En plus de la tokenisation, une solution HCE nécessite d'autres mesures de sécurité dont :

Vérification du système d'exploitation. Cette propriété vise à détecter les appareils rootés. En effet, comme une application malveillante peut rouler avec les privilège *root* dans le cas d'un système rooté, le risque de récupération des clés cryptographiques est plus élevé. Android permet différentes méthodes de vérification que les développeurs peuvent utiliser telles que la vérification du programme `su` qui permet d'accorder des privilèges root ou l'exécution de commande en tant que root sans contrainte sur un système rooté, ou la vérification du programme *BusyBox* qui est disponible sur la majorité des ROM personnalisées.

Module logiciel *tamper-proof*. Le but ici est d'empêcher un attaquant de modifier le code de l'application statiquement ou dynamiquement. Cette propriété est généralement validée par différentes techniques telles que la vérification de l'intégrité de l'application à l'exécution, ou aussi l'obfuscation du code de l'application pour entraver un attaquant effectuant une rétro-ingénierie.

Profilage de l'appareil. Cette technique consiste à recueillir des informations de l'appareil depuis lequel un utilisateur a effectué son inscription de façon à construire une empreinte du système. De ce fait, un attaquant qui a réussi à obtenir un jeton de service et essayant d'obtenir accès au service à partir d'un autre appareil sera détecté.

Authentification biométrique. Cette fonctionnalité consiste à ajouter un autre facteur d'authentification par biométrie (quelque chose que l'on est) telle que l'empreinte digitale et la reconnaissance faciale. En effet, la solution Android Pay propose l'empreinte digitale comme option additionnelle pour sécuriser les transactions.

Utilisation de stockage matériel. Ce cas est l'exemple d'une solution hybride où le mode HCE peut utiliser un support matériel pour stocker les *jetons* tels que un TEE ou même un SE.

Tableau 3.1 Comparaison entre solution HCE et SE

Propriétés	Solution secure element SE	Solution logicielle HCE
Exigence de l'appareil mobile	Appareil avec NFC. Carte SIM. Protocole de communication entre le SE et le contrôleur NFC. Certification de l'appareil et du fournisseur de service et du fournisseur du SE.	Appareil avec NFC. Android 4.4 et +.
Provisionnement	Nécessite un TSM pour le déploiement de services et des secrets sur le mobile. Changer d'appareil mobile ou de SE (en cas de SIM) implique la perte du service, l'utilisateur doit se réinscrire de nouveau.	Un simple téléchargement du magasin d'application suffit pour bénéficier du service.
Sécurité	Clés cryptographiques et données secrètes sont stockées sur une mémoire matérielle inviolable et isolée du système d'exploitation	Plusieurs options sont disponibles : tokenisation, solutions hybrides.
Modèle d'affaire	Nécessite la collaboration avec les fournisseurs des SE ainsi que le TSM. Assez complexe à gérer et plus coûteux à maintenir.	Modèle moins complexe, le fournisseur de service n'a pas besoin d'établir une relation avec un TSM.

3.4 Conclusion

Le système d'exploitation mobile Android dispose de différentes solutions de stockage de clés cryptographiques dans un contexte d'authentification NFC. Ces solutions varient d'une approche purement logicielle, facile à maintenir à une solution purement matérielle plus sécuritaire mais aussi plus complexe et impliquant autres intervenants tel que le TSM et le fournisseur de matériel de stockage, à une solution hybride combinant un service HCE à un support de stockage isolé tel que le TEE. L'adoption de l'une ou l'autre reste un choix du fournisseur de service, ce choix a été initialement limité par la disponibilité de la technologie sur mobile poussant à une solution SE malgré sa complexité. Cependant, l'apparition de HCE sur Android a commencé à faire migrer le marché vers une solution hybride, ou voire

même purement logicielle, facilitant l'intégration des solutions NFC avec un moindre délai de commercialisation. Maintenant que nous avons discuté les options disponibles pour sécuriser les clés cryptographiques d'une application d'authentification sur mobile, nous discuterons dans le chapitre suivant différentes implémentations d'algorithmes cryptographiques à clé publique et leur viabilité en termes de temps d'exécution.

CHAPITRE 4

PROTOCOLE D'AUTHENTIFICATION MUTUELLE BASÉ SUR LA CRYPTOGRAPHIE À COURBE ELLIPTIQUE

4.1 Introduction

Comme discuté au chapitre 2, les applications RFID conçues initialement à des fins d'identification présentent plusieurs faiblesses de sécurité. En effet, fondées principalement sur des algorithmes cryptographiques symétriques et propriétaires, aucune des solutions RFID ou NFC existantes ne proposent d'algorithme d'authentification assurant la propriété de non-clonabilité, définie au chapitre 1. L'authentification complète dans ce contexte le processus d'identification dans le sens ou l'authentification permet de confirmer l'identité du porteur de carte. Dans un système de contrôle d'accès, un processus de contrôle valide l'identité et, après authentification, donne l'accès ou le refuse dans le cas contraire.

Nous nous intéressons dans ce chapitre à l'authentification NFC sur mobile, et nous proposons un protocole d'authentification NFC basé sur la cryptographie à courbe elliptique qui répond aux attaques de clonage de carte.

Malgré le fait que les téléphones intelligents soient équipés de ressources et de capacités de calcul comparables à celles des ordinateurs, nous avons opté pour une solution qui respecte les contraintes de ressource d'un système embarqué pour que notre protocole soit facile à porter sur une carte RFID avec des ressources limitées. Pour cela, la cryptographie à courbe elliptique est le choix parfait pour résoudre le problème de distribution de clés et assurer le niveau de sécurité désiré avec un minimum de capacité de calcul.

4.2 Protocole proposé

Nous proposons un protocole d'authentification utilisant une infrastructure à clé publique. L'approche ICP est une approche sécurisée pour distribuer les clés qui nous permet de garantir l'authentification de la carte et du lecteur grâce aux certificats à clé publique. Elle élimine entièrement la nécessité de garder une copie de la clé secrète dans le lecteur NFC. Notre analyse montre que les signatures à courbe elliptique fournissent une gestion de la mémoire plus efficace pour des environnements à contraintes fortes tels que les étiquettes NFC et les appareils mobiles par une grande marge par rapport aux solutions équivalentes fondées sur RSA.

Pour cela, nous avons opté pour une solution d'authentification utilisant un mécanisme de signature par courbe elliptique ECDSA qui est inspiré de l'algorithme classique de signature DSA. Nous avons aussi utilisé un mécanisme échange de clés par courbe elliptique à la manière de Diffie et Hellman.

Voici le séquencement des opérations lorsque l'utilisateur demande un accès auprès du fournisseur de service. Le fournisseur de service possède un certificat délivré par une autorité de certification racine qui lui permettra de s'authentifier auprès de la carte. Après vérification de légitimité du fournisseur de service, ce dernier ajoute la carte à la base de donnée de contrôle d'accès selon les droits qui lui seront autorisés et lui délivre un certificat qui permettra au lecteur d'authentifier la carte.

4.2.1 Entités

Le protocole que nous proposons implique quatre types d'entités :

Carte. Cette entité joue le rôle habituel d'une carte NFC avec une capacité additionnelle de calcul cryptographique.

Lecteur. Accomplit le rôle habituel de lecteur, permettant de communiquer avec une carte NFC et une infrastructure de contrôle d'accès (*back-end*). Ce lecteur permet non seulement d'identifier et authentifier une carte présente à champ de communication, mais permet aussi à cette même carte de l'authentifier. Cette approche permet aux cartes de ne divulguer leur informations qu'à un lecteur certifié.

Unité de déploiement. L'unité de déploiement est responsable de la certification de différents lecteurs et cartes composant le système. Cette unité peut classer les cartes/lecteurs par service : nous pouvons imaginer que le même fournisseur peut offrir différents services comme le contrôle d'accès, la billetterie ou autres, pour cela, une paire de clés de signature sera générée pour chaque service. Dans ce cas, un même téléphone, adhérant à deux services différents, aura deux certificats différents générés par deux clés de signature différentes.

Unité d'infrastructure. Cette entité interagit uniquement avec l'unité de déploiement pour la certifier. Toute autre entité reçoit la clé publique de l'infrastructure pour pouvoir vérifier le certificat de l'unité de déploiement. De ce fait, un processus permettant de pousser cette clé à distance sera nécessaire tel que la technologie OTA.

4.2.2 Phase d'initialisation

Dans cette phase d'initialisation, chaque entité disposant de ses clés de signature ECDSA stockées chiffrées sur le mobile, commence par récupérer ces dernières clés et générer par la

suite une deuxième paire de clés ECDH qui ne sera pas stockée sur le dispositif et sert à créer un secret partagé par échange de clés. Ensuite, ce secret partagé permettra d'établir un canal sécurisé utilisant un chiffrement symétrique tel que AES. Dans cette phase, l'unité de déploiement génère trois paires de clés ECDSA, une première paire dont la clé publique sera signée par l'unité d'infrastructure, les deux autres paires servent à signer respectivement les clés de cartes et lecteurs. Cette approche de séparation du chemin de signature de clé nous permettra de garantir la souveraineté des clés : si une clé est compromise on n'aura pas à changer toutes les entités. Une entité A, suivant son type, effectue les opérations suivantes lors de la phase d'initialisation :

1. Choisit une courbe elliptique E de domaine $D = (p, a, b, G, n, h)$
2. Génère une paire de clés ECDSA (Q, d) telle que $d \in [1, p-1]$ et $Q = d * G$.
3. Génère une paire de clés ECDH $(k * G, k)$ telle que $k \in [1, p-1]$.

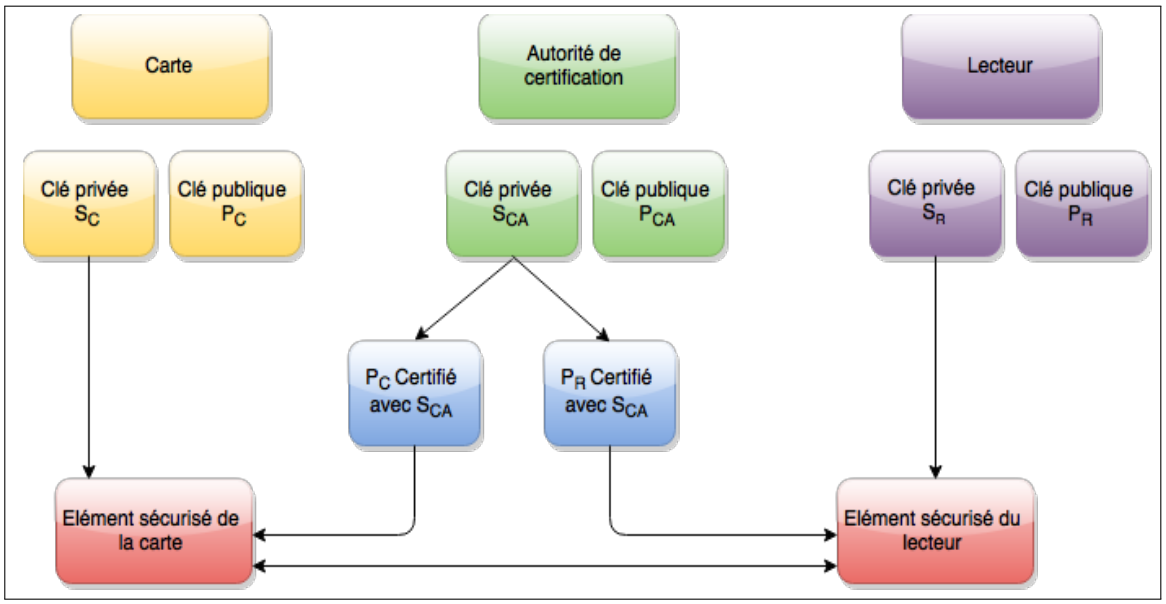

Figure 4.1 Initialisation

4.2.3 Déploiement

Dans cette phase, les différentes unités installent leur application correspondante, choisissent leur code NIP qui sert pour le déverrouillage de l'application ainsi que pour le chiffrement des fichiers sur le disque, génèrent leurs clés ECDSA suivant des paramètres de courbe elliptique fixe et les stockent sur le dispositif NFC. Ensuite, les entités de certification se chargent de signer ces derniers clés : l'infrastructure certifiant l'unité de déploiement et cette dernière certifiant les cartes et lecteurs.

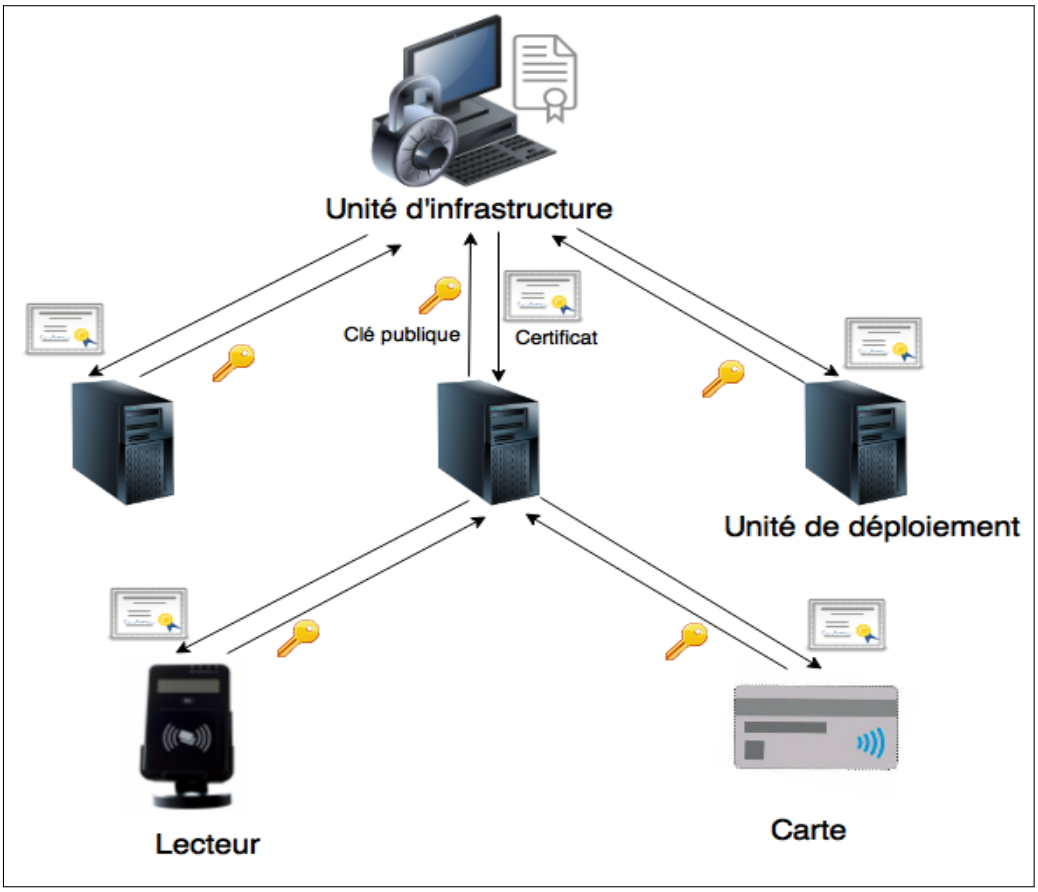

Figure 4.2 Déploiement

4.2.4 Phase d'authentification

Cette phase commence par un échange de clés Diffie et Hellman à courbes elliptiques. Ensuite, un canal sécurisé est établi permettant aux dispositifs communiquant de partager des données en sécurité. Un mécanisme d'authentification par rôle grâce à un identifiant *NodeType* permet de différencier les dispositifs suivant leurs rôles. En effet, un lecteur ou une carte, après établissement d'un canal sécurisé, authentifie l'unité de déploiement par son certificat livré par l'unité d'infrastructure. Une authentification lecteur-carte se fait grâce aux certificats délivrés par l'unité de déploiement.

Authentification de l'unité d'infrastructure et de l'unité de déploiement

Cette étape est préliminaire aux autres étapes de déploiement de la solution. Pour qu'un lecteur et une carte puissent s'authentifier en utilisant ce protocole, ils doivent être certifiés par une unité de déploiement légitime commune. Cette dernière tire sa légitimité de cette phase de certification.

1. L'unité de déploiement envoie à l'unité d'infrastructure $K_D * G$ ainsi que son numéro de version *VersionNumber* qui indique les paramètres de sa courbe elliptique.
2. L'unité d'infrastructure vérifie que l'unité de déploiement dispose des mêmes paramètres de courbe elliptique. Elle calcule ensuite le secret partagé $S = K_I * K_C * G$ et envoie à l'unité de déploiement $K_I * G$.
3. L'unité de déploiement calcule à son tour le secret partagé $S = K_D * K_I * G$ ainsi que la signature (r, s) de ce dernier, tel que $r = (G * k)_x \mod n$, $s = k^{-1} * (\text{SHA-1}(S) + d_D * r) \mod n$, où k est un nombre aléatoire $k \in [1, n-1]$. Cette unité envoie ensuite une réponse chiffrée avec l'algorithme symétrique AES avec clé secrète S, composée de la signature $(r, s)_S$, la clé publique ECDSA Q_D et l'identifiant de rôle *NodeType*.
4. L'unité d'infrastructure déchiffre la réponse reçue en utilisant le secret S calculé à l'étape 2. Ensuite, elle vérifie la signature $(r, s)_S$ avec la clé Q_D et calcule la signature ECDSA $(r, s)_{Q_D}$, tel que $r = (G * p)_x \mod n$, $s = p^{-1} * (\text{SHA-1}(Q_D) + d_I * r) \mod n$, où p est un nombre aléatoire $p \in [1, n-1]$. Enfin, l'unité d'infrastructure envoie à l'unité de déploiement son identifiant type *NodeType* et la signature $(r, s)_{Q_D}$.

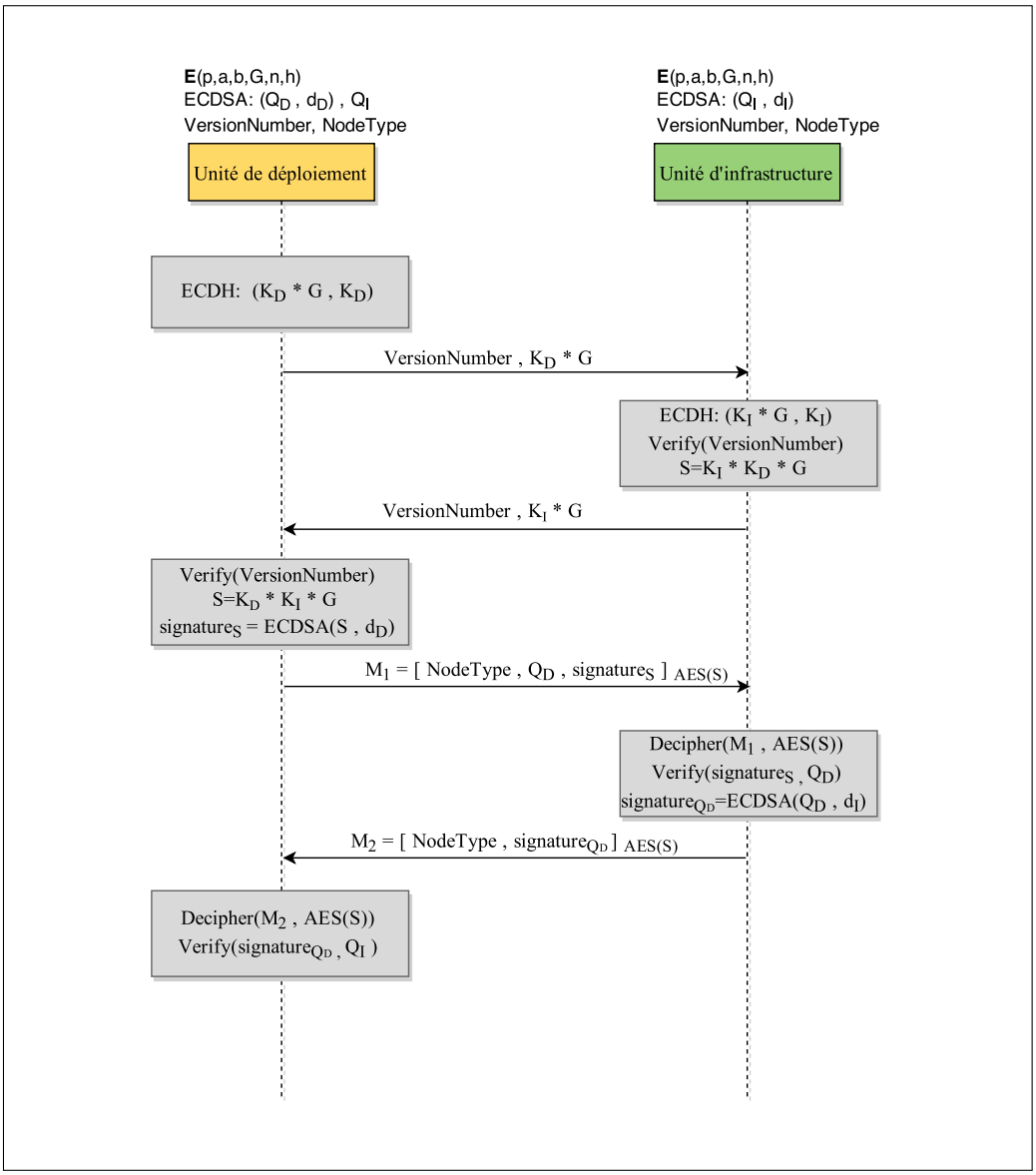

Figure 4.3 Certification de l'unité de déploiement

Authentification de l'unité de déploiement et du lecteur

Cette étape permet à une unité de déploiement certifiée par l'unité d'infrastructure de certifier un lecteur appartenant au système. Après une initialisation réussie, un lecteur et une unité de déploiement présents dans un champs de communication NFC :

1. Le lecteur commence l'établissement d'un canal sécurisé en envoyant à l'unité de déploiement $K_R * G$ ainsi que son numéro de version *VersionNumber* indiquant les paramètres de sa courbe elliptique.
2. L'unité de déploiement vérifie que le lecteur dispose des mêmes paramètres de courbe

elliptique. Ensuite elle calcule le secret partagé $S = K_D * K_R * G$ et envoie au lecteur $K_D * G$.

3. Le lecteur termine l'établissement de canal sécurisé en calculant le secret partagé $S = K_R * K_D * G$. Il calcule ensuite ainsi la signature $(r, s)_S$, tel que $r = (G * k)_x \mod n$, $s = k^{-1} * (\text{SHA-1}(S) + d_R * r) \mod n$, où k est un nombre aléatoire $k \in [1, n-1]$. Le lecteur envoie cette dernière signature $(r, s)_S$, sa clé publique ECDSA Q_R et l'identifiant de rôle *NodeType* à l'unité de déploiement.

4. L'unité de déploiement commence par déchiffrer la réponse reçue en utilisant le secret S calculé à l'étape 2. Ensuite, elle vérifie la signature $(r, s)_S$ avec la clé Q_R et calcule la signature ECDSA $(r, s)_{Q_R}$ tel que $r = (G * p)_x \mod n$, $s = p^{-1} * (\text{SHA-1}(Q_R) + d_D * r) \mod n$, ou p est un nombre aléatoire $p \in [1, n-1]$. Enfin, elle envoie au lecteur son identifiant de type *NodeType*, la signature $(r, s)_{Q_R}$, son certificat $(r, s)_{Q_D}$ et la clé de vérification de certificat des cartes Q_{D*} chiffrés en utilisant le secret partagé avec l'algorithme AES.

5. Le lecteur déchiffre le message reçu et valide le certificat $(r, s)_{Q_D}$ en utilisant la clé de vérification de signature de l'unité d'infrastructure Q_{I*}.

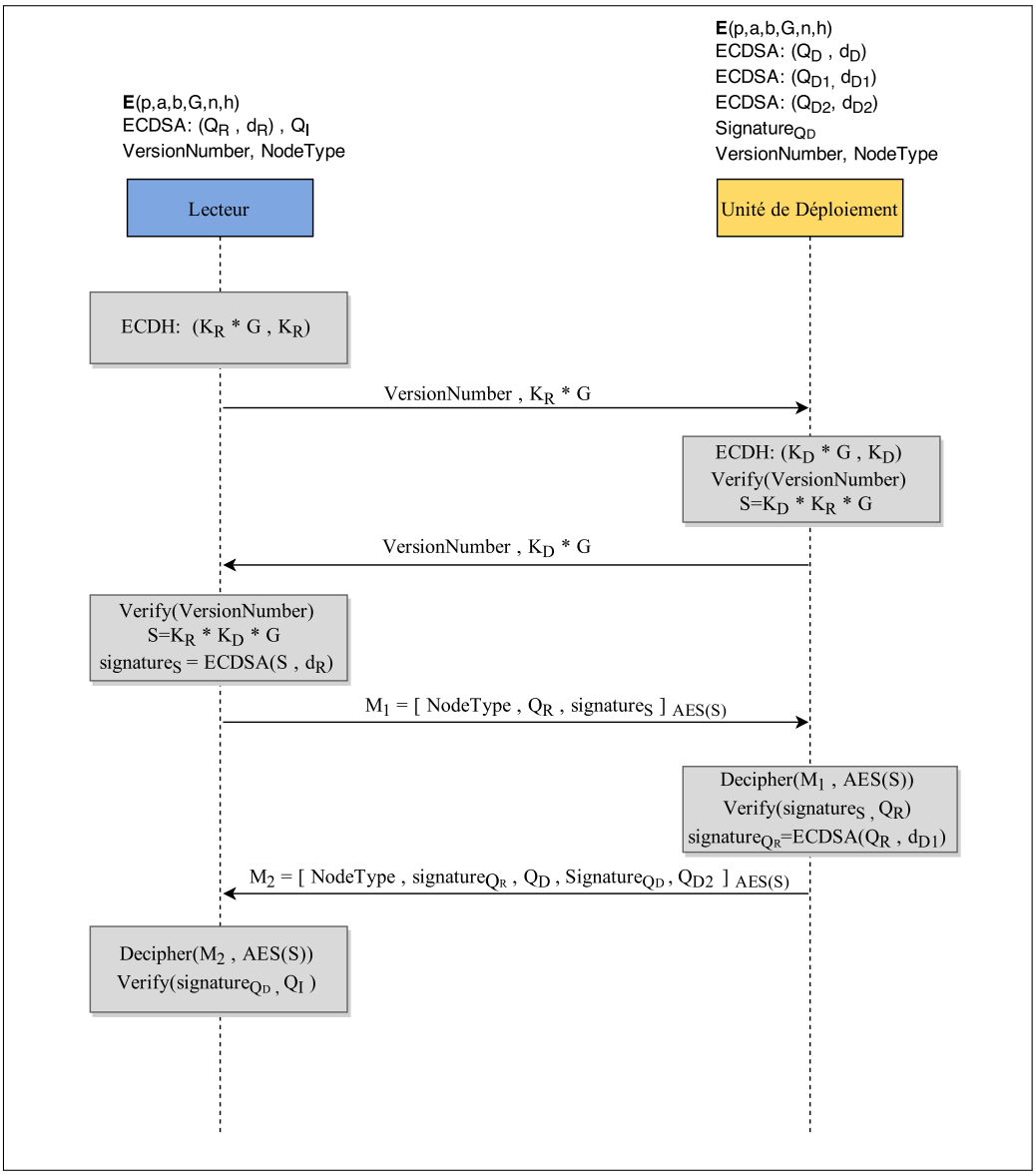

Figure 4.4 Certification du lecteur

Authentification de l'unité de déploiement et de la carte

De même que dans l'étape prétendante, l'unité de déploiement certifiée par l'unité d'infrastructure de s'authentifier à une carte et la certifie comme suit :

1. L'unité de déploiement envoie à la carte $K_D * G$ ainsi que son numéro de version *VersionNumber* indiquant les paramètres de sa courbe elliptique.
2. La carte vérifie que l'unité de déploiement dispose des mêmes paramètres de courbe elliptique. Ensuite elle calcule le secret partagé $S = K_C * K_D * G$ et envoie à l'unité de déploiement $K_C * G$.

3. L'unité de déploiement calcule à son tour le secret partagé $S = K_D * K_C * G$. Ensuite, elle calcule la signature $(r, s)_S$, tel que $r = (G*k)_x \mod n$, $s = k^{-1}*(\text{SHA-1}(S)+d_D*r) \mod n$, où k est un nombre aléatoire $k \in [1, n-1]$. L'unité de déploiement envoie la signature $(r, s)_S$, sa clé publique ECDSA Q_D et l'identifiant de rôle *NodeType* à la carte.
4. Après déchiffrement du message reçu à l'aide de la clé S calculée dans l'étape 2, la carte vérifie la signature $(r, s)_S$ avec la clé Q_D et renvoie à son tour la signature $(r, s)_S$, la clé publique Q_C et l'identifiant *NodeType* tout en gardant le canal sécurisé.
5. L'unité de déploiement commence par déchiffrer la réponse reçue en utilisant le secret partagé S. Ensuite, elle vérifie la signature $(r, s)_S$ avec la clé Q_C et calcule la signature ECDSA $(r, s)_{Q_C}$ tel que $r = (G*p)_x \mod n$, $s = p^{-1}*(SHA-1(Q_C)+d_D*r) \mod n$, où p est un nombre aléatoire $p \in [1, n-1]$. Puis, elle envoie à la carte son identifiant type *NodeType*, la signature $(r, s)_{Q_C}$, son certificat $(r, s)_{Q_D}$ et la clé de vérification de certificat des lecteurs Q_{D*} chiffrés en utilisant le secret partagé avec l'algorithme AES.
6. La carte déchiffre le message reçu et valide le certificat $(r, s)_{Q_D}$ en utilisant la clé de vérification de signature de l'unité d'infrastructure Q_{I*}

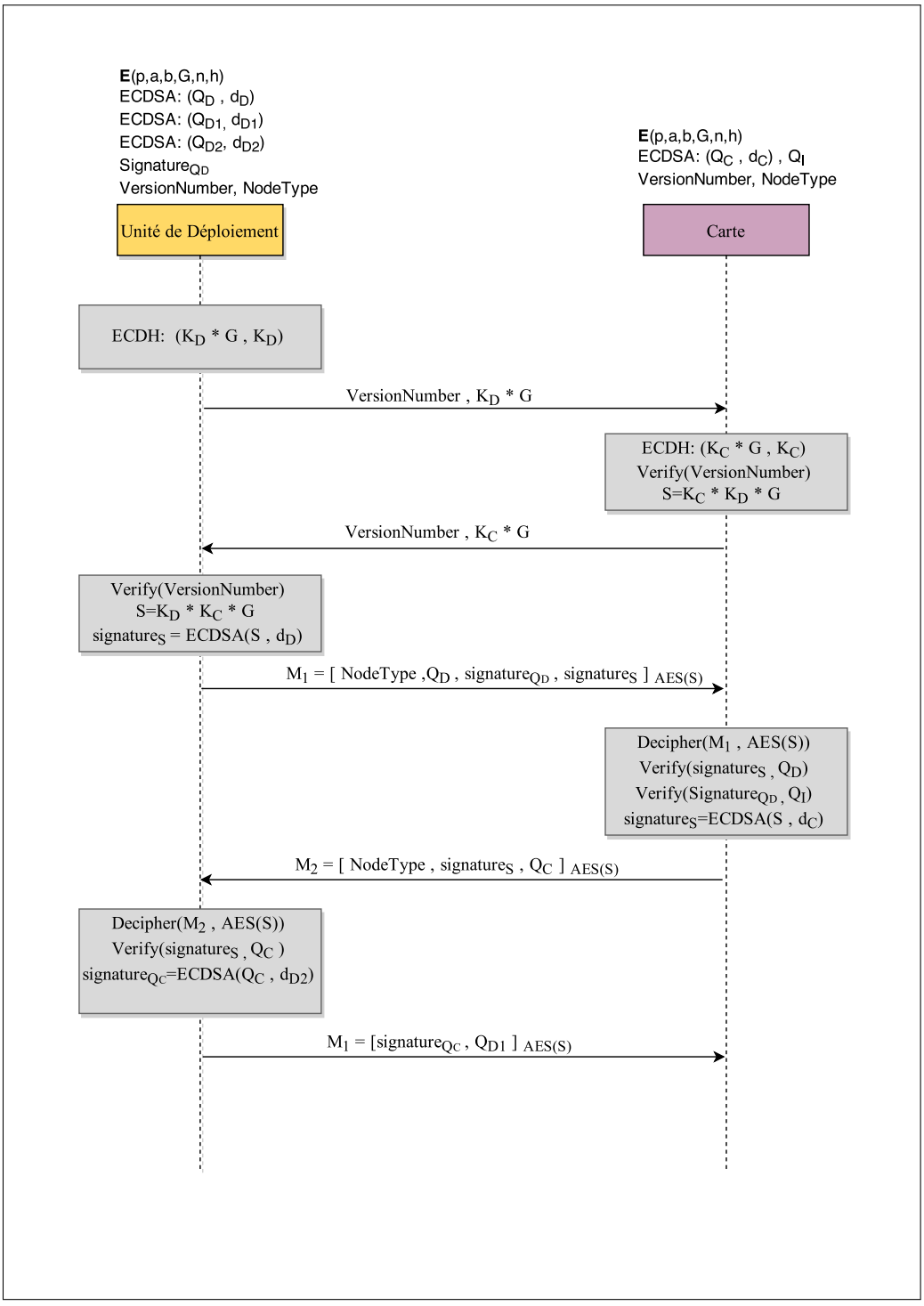

Figure 4.5 Certification de la carte

Authentification du lecteur et de la carte

Un lecteur et une carte certifiés de la même unité de déploiement, après avoir passé par une phase d'initialisation et présents dans un champ de communication NFC peuvent compléter leur authentification mutuelle comme suit :

1. Le lecteur envoie à la carte $K_R * G$ ainsi que son numéro de version *VersionNumber* qui indique les paramètres de sa courbe elliptique.
2. La carte vérifie que le lecteur dispose de mêmes paramètres de courbe elliptique. Ensuite, elle calcule $S = K_C * K_R * G$ et envoie au lecteur $K_C * G$.
3. Le lecteur calcule aussi le secret partagé $S = K_R * K_C * G$ ainsi que la signature (r, s) du secret partagé S, tel que $r = (G * k)_x \mod n$, $s = k^{-1} * (\text{SHA-1}(S) + d_R * r) \mod n$, où k est un nombre aléatoire $k \in [1, n-1]$. Ensuite, le lecteur envoie la signature $(r, s)_S$, son certificat $(r, s)_{Q_R}$, sa clé publique ECDSA Q_R ainsi que son identifiant de rôle *NodeType*. Le message envoyé est chiffré avec l'algorithme symétrique AES de clé secrète S.
4. La carte commence par déchiffrer le message reçu en utilisant le secret S calculé à l'étape 2. Maintenant, possédant la clé de vérification de la signature $(r, s)_{Q_R}$, la carte peut vérifier qu'elle communique bien avec un lecteur légitime (certifié). Elle vérifie aussi la signature $(r, s)_S$ avec la clé publique Q_R. A son tour, la carte calcule (r, s) du secret partagé S, tel que $r = (G * p)_x \mod n$, $s = p^{-1} * (\text{SHA-1}(S) + d_C * r) \mod n$, où p est un nombre aléatoire $p \in [1, n-1]$. Ensuite, elle envoie au lecteur la signature $(r, s)_S$, son certificat $(r, s)_{Q_C}$, sa clé publique ECDSA Q_C ainsi que son identifiant de rôle *NodeType*. Le canal sécurisé est toujours maintenu grâce au chiffrement AES avec la clé partagée.
5. Le lecteur déchiffre de la même façon le message reçu. Ensuite, il vérifie la signature $(r, s)_{Q_C}$ avec la clé de vérification reçue de l'unité de déploiement et vérifie la signature $(r, s)_S$ avec la clé Q_C.

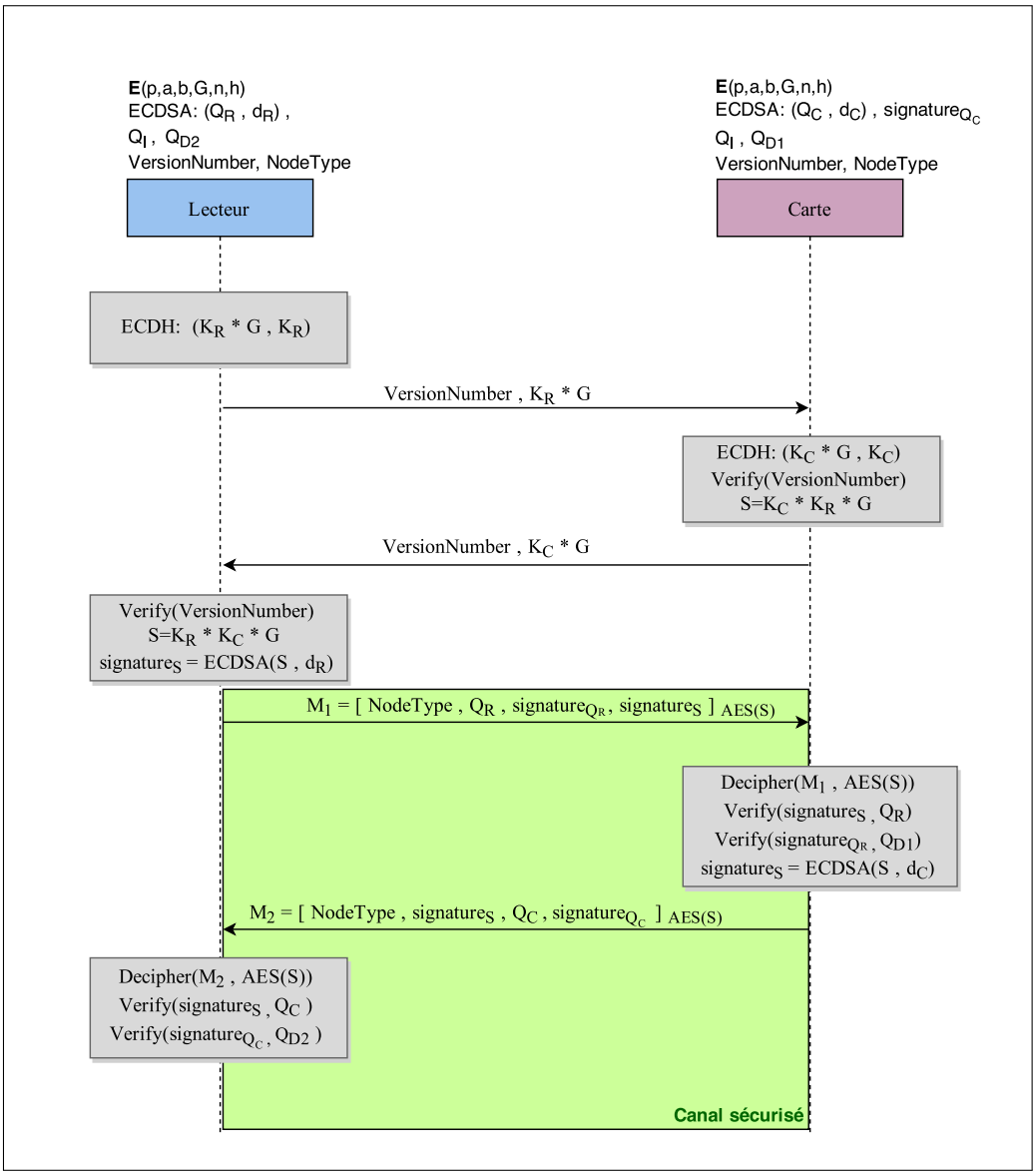

Figure 4.6 Authentification lecteur carte

4.3 Conclusion

Nous avons proposé dans ce chapitre un protocole d'authentification NFC basé sur une infrastructure à clé publique ICP. Nous avons utilisé pour ce protocole un cryptosystème à clé publique et courbe elliptique. Nous avons détaillé les différentes étapes du protocole ainsi que les différentes entités permettant le déploiement et la vérification de légitimité des lecteurs et cartes. Nous étudions dans le chapitre suivant la viabilité de ce système en termes de temps d'exécution.

CHAPITRE 5

ÉTUDE EXPÉRIMENTALE

Les accès sécurisés présentent des enjeux importants, en particulier dans certains types d'établissements, à commencer par les sites militaires et industriels à risques, les aéroports, les tertiaires exigeants un haut degré de sécurité comme les banques pour l'accès du personnel, les sièges sociaux de grandes entreprises sans oublier les universités et les ministères. La solution clé pour protéger l'échange des données dans la communication est la cryptographie.

Comme le niveau de sécurité d'une identification NFC ne doit pas nuire à la fluidité des accès, il faut donc atteindre des temps de lecture acceptables inférieurs à 1 seconde (s) au-delà desquels un accès est considéré comme « ralenti ». Nous étudions ainsi dans ce chapitre la viabilité en termes de temps d'exécution de différentes implémentations du protocole proposé au chapitre 4. Nous détaillons ainsi les différents paramètres de ces implémentations soit les environnements d'exécution, le choix de mode d'opération NFC et le choix des paramètres de la courbe elliptique.

Afin d'évaluer la performance de ce type solution selon différents scénarios d'utilisation possibles dans l'authentification par NFC, nous avons conçu trois expériences types, où différentes versions de ces protocoles ont été implémentées :

1. Le premier protocole n'implémente que de l'authentification de la carte vers le serveur. Aucun chiffrement entre le carte et le lecteur n'est implémenté. Ceci correspondrait à un scénario typique d'utilisation dans les transports en commun.
2. Le deuxième protocole implémente une authentification mutuelle entre la carte et le lecteur. Aucun chiffrement des communications subséquents n'est implémenté. Ce scénario correspondrait à une application de plus haute sécurité, tel que le contrôle d'accès physique ou paiement, où les informations subséquentes (code d'autorisation, identificateur de cartes) ne sont pas sensibles.
3. Le troisième protocole implémente une authentification mutuelle et un chiffrement de session pour les échanges subséquents. Ce scénario pire cas correspond à une application où pour des raisons de rétro-compatibilité l'identifiant de carte transmis doit être protégé car il peut être utilisé pour pénétrer autrement le système de contrôle d'accès.

5.1 Critères de performance

Dans ces trois cas, nous étudions la performance en termes de temps d'exécution d'une session d'authentification, y compris l'établissement de la clé de chiffrement, le cas échéant. Cependant, nous ne tenons pas compte des temps d'exécution des phases d'initialisation et de déploiement, comme nous jugeons ce temps non critique. En effet, afin de faciliter nos tests, nous avons opté pour une solution mobile pour les entités unité de déploiement et unité d'infrastructure, mais nous pouvons bien imaginer un autre type de solution et voire même utiliser une autre technologie que le NFC pour déployer les certificats.

Pour chacun de ces protocoles, nous avons néanmoins évalué les temps de génération de clés pour les différentes versions des protocoles d'authentification proposés, en respectant toujours le même principe d'utilisation dICP pour authentifier une carte présente devant un lecteur NFC.

Tableau 5.1 Temps d'exécution expérimental de génération de clés cryptographiques pour différents paramètres de domaine de la courbe elliptique en ms (sur un Nexus 5).

Opération	P-192	P-224	P-256	P-384	P-521
Génération de clés ECDSA	329	356	483	700	793

Tel qu'on peut le constater au tableau 5.1, ces temps sont effectivement acceptables lors d'une phase d'enrôlement ou d'enregistrement.

Pour évaluer expérimentalement nos implémentations du protocole d'authentification, nous avons réalisé des séries de 20 tests, notamment la génération des clés publiques et privées, la signature et la vérification de signature ECDSA et l'échange de clés ECDH.

Nous évaluons la performance du temps d'exécution en comparaison avec une contrainte de 1 s que doit respecter notre implémentation du protocole proposé au chapitre 4. Nous définissons ainsi le temps expérimental d'authentification comme la moyenne des 20 tests effectués. Tous les écart types des tests effectués sont entre 5% et 10%.

Comme le protocole que nous évaluons repose principalement sur des opérations de signature et de vérification de signature ECDSA, il est nécessaire de valider que le temps de calcul de ces opération ne dépasse pas le seuil établi de 1s. Tel qu'illustré au tableau 5.2, le temps de calcul des opération ECDSA est inférieur à 1s pour la majorité des paramètres elliptiques utilisés.

Pour modéliser le temps de transmission NFC nous nous référons au débit de transfert NFC sur Android qui est de 424 kbps. Nous calculons donc le temps de transmission NFC théorique selon la quantité d'informations échangée entre les deux dispositifs NFC carte et lecteur. Ce temps calculé théoriquement diffère de celui mesuré expérimentalement puisque

Tableau 5.2 Temps d'exécution d'opération de signature et vérification de signature ECDSA d'un objet de même taille de clé pour différents paramètres de domaine de la courbe elliptique en ms (sur un Nexus 5).

Opération	P-192	P-224	P-256	P-384	P-521
Signature ECDSA	182	205	291	448	620
Vérification de signature ECDSA	244	316	374	535	890

le temps de transmission NFC expérimental comprend un traitement effectué par le système d'exploitation pour gérer le service de communication NFC.

Pour modéliser le temps d'exécution, nous avons aussi besoin de calculer la taille des objets cryptographiques échangés, notamment la signature ECDSA, qui constitue l'élément fondamental du protocole proposé. En effet, le couple d'entiers (r, s) constituant la signature ECDSA, tel que nous l'avons définie dans la section 2.4.5, est de la taille de la courbe elliptique. De ce fait, une signature ECDSA de l'environnement P-256 sera de longueur 2*256 bits. Nous envoyons de plus une entête comprenant un identifiant d'objet *Header* et la taille de l'objet. Nous pouvons donc modéliser le temps d'exécution comme suit :

$$T_{exec} = T_{trans_{NFC}} + T_{calcul}$$

tel que T_{calcul} comprend les opérations de signature ECDSA, de vérification de signature ECDSA et d'échange de clés ECDH, et le temps de transmission total incluant le temps de traitement NFC du système d'exploitation

$$T_{trans_{NFC}} = \sum_{objet} T_{trans_{NFC}}(objet) + T_{traitementNFC} = \sum_{object} \frac{\text{length}(objet)}{\text{debit}_{NFC}} + T_{traitement_{NFC}}$$

où la longueur de chaque objet transmis dépend de l'entête de type, de l'entête de longueur et de la longueur de la charge utile de l'objet lui-même :

$$\text{length}(objet) = \text{HeaderBytes}(objet) + \text{LengthBytes}(objet) + \text{ObjectBytes}(objet)$$

5.2 Environnement de test

Plusieurs choix technologiques d'implémentation du protocole proposé étaient possibles pour cette évaluation expérimentale. Nous avons donc fait un choix entre systèmes d'exploitation mobile (Android, Blackberry, iOS, Windows Phone), mode d'opération NFC (lecteur/carte, pair à pair, émulation de Carte) et technologie de stockage de secret cryptographique (élément de sécurité : embarqué dans le mobile, carte SIM ou entièrement logiciel).

Nous faisons dans ce qui suit la discussion de nos choix technologiques soit un système Android Kitkat(Version 4.4) avec la technologie d'émulation de carte NFC pour les entités de types carte et le mode d'opération lecteur pour les lecteurs de notre protocole. Ces choix ont été conditionnés par les contraintes d'évaluation dans un contexte de laboratoire, qui ne seraient normalement pas présentes dans une déploiement commercial. Ceci justifie, par exemple, le stockage de clés en mémoire plutôt que les autres solutions qui nécessitent un partenariat avec les fabricants de plateforme mobiles ou les opérateurs de télécommunications.

Système d'exploitation mobile. Pour décider de la plateforme mobile de développement, nous avons commencé par éliminer iOS de notre liste de choix comme le système d'exploitation ne disposait pas de la technologie NFC quand nous avons commencé notre étude. Bien que ce géant mobile a introduit cette technologie à sa plateforme durant le dernier trimestre 2015, la fonctionnalité est limitée au support de leur solution de paiement mobile *Apple Pay* et leur *SDK* NFC n'est pas encore disponible aux développeurs. Il en est de même pour le système d'exploitation mobile Windows Phone qui ne disposait pas de la fonctionnalité d'émulation de carte au moment où nous avons commencé ce travail. De ce fait, nous avons choisi Android comme plateforme de développement du prototype. En effet, Android est la plateforme la plus répandue sur le marché avec une multitude de modèles d'appareils supportant la technologie NFC.

Mode d'opération NFC. Nous étions bloqués par des limitations de la technologie au début de nos recherches. En effet, le mode émulation de carte, nécessaire à l'implémentation de la carte, n'était pas disponible sur les versions officielles d'Android. Nous avons tenté ensuite le mode pair à pair qui nous a aussi limité puisque Android exige de passer par son application Beam pour échanger des messages NFC. L'arrivée de la fonctionnalité HCE au dernier trimestre de 2014 nous a facilité la tâche d'implémentation de la carte.

Bien que les éléments sécurisés matériels, embarqués sur le téléphone mobile ou sur la carte SIM, constituent un environnement isolé, inviolable offrant le plus de sécurité possible aux services NFC, ils présentent plusieurs contraintes en termes de facilité de déploiement. Aujourd'hui, c'est un tiers de confiance, le TSM, qui va pouvoir installer des application sur le SE. De plus, c'est lui qui va établir le lien entre un fournisseur de services et l'opérateur téléphonique tout en garantissant la sécurité des informations sensibles de l'usager. On réalise donc le nombre important d'interlocuteurs qui agissent dans une telle solution NFC se basant sur la communication avec un élément de sécurité : il faudra donc, comme indiqué précédemment, collaborer avec les opérateurs télécomms pour qu'ils intègrent des implémentations du protocole SWP dans leurs versions personnalisées de système d'exploitation Android pour

pouvoir communiquer avec la carte SIM, développer une application Android native ainsi qu'une application javacard qui sera déployée par le TSM sur la carte SIM.

Plateforme mobile Nous avons utilisé deux appareils Nexus 5, un Samsung Galaxy S3 et une tablette Nexus 7 pour nos tests. La version d'Android utilisée était la *KitKat* 4.4, que nous avons sélectionnée afin d'avoir la fonctionnalité d'émulation de carte logicielle HCE.

Figure 5.1 Plateformes mobiles utilisées dans nos tests

Librairie Cryptographique. Nous avons utilisé la librairie cryptographique mobile *SpongyCastle* qui est l'équivalent de la librairie *BouncyCastle* sur le système Android. Il s'agit d'une librairie d'outils cryptographiques de code source libre pour Java, comparable à OpenSSL en C. Cette librairie comporte une multitude d'algorithmes cryptographiques notamment les algorithmes de ECC non disponible sur l'*API* de cryptographie sur Android. *SpongyCastle* nous permet de choisir les paramètres de la courbe elliptique selon la norme *National Institute of Standards and Technology* (NIST).

Tableau 5.3 Spécification NIST des paramètres des courbes elliptiques

Spécification	Taille clé	Taille signature
P-192	192	384
P-224	224	448
P-256	256	512
P-384	384	768
P-521	521	1042

Cette librairie a été utilisée pour l'implémentation des protocoles d'authentification décrits au chapitre 4. De plus, dans une solution classique de contrôle d'accès, la phase d'authentification de la carte et lecteur est suivie généralement d'un envoi de secret tel que l'identifiant de la carte. Il est donc très important de sécuriser aussi cette dernière étape de transfert de secret. Pour le faire, nous proposons un mécanisme d'échange de clés par courbe elliptique à la manière de Diffie et Hellman. Il s'agit d'un échange de clés sans interaction sur un canal non sécurisé en utilisant ECC et qui permet aux cartes et lecteurs d'établir un canal sécurisé.

Stockage des clés cryptographiques. Étant donné que nous n'avions pas d'ententes particulières, ni avec les opérateurs de télécommunications, ni avec les fabricants de plateformes mobiles, il nous était impossible d'avoir accès à un SE matériel, qu'il soit implémenté sur une carte SIM ou sur SE embarqué. Nous avons donc opté pour une solution de SE logiciel, mettant de côté son évaluation du point de vue de la sécurité. Ceci ne compromet pas nos résultats de performance, car il est prévisible que les calculs cryptographiques réalisés sur des SE matériel soient au moins aussi rapides.

Comme nous avons opté pour une solution purement logicielle, la solution disponible pour notre prototype et dans le contexte d'une solution de contrôle d'accès d'une petite et moyenne entreprise est un stockage logiciel interne. Cependant, nous avons ajouté des couches de protection de ces données sensibles par chiffrement de fichier contenant les clés cryptographiques.

5.3 1er protocole : Authentification de carte

Dans ce premier test, tel que décrit dans la figure 5.2, une carte s'authentifie auprès d'un lecteur tous deux certifiés par la même unité de déploiement. Il s'agit d'une simple authentification de la carte par un mécanisme de défi/réponse où le lecteur vérifie qu'il s'agit bien d'une carte certifiée par la même unité de déploiement. Cependant, la carte ne vérifie pas dans ce cas la légitimité du lecteur.

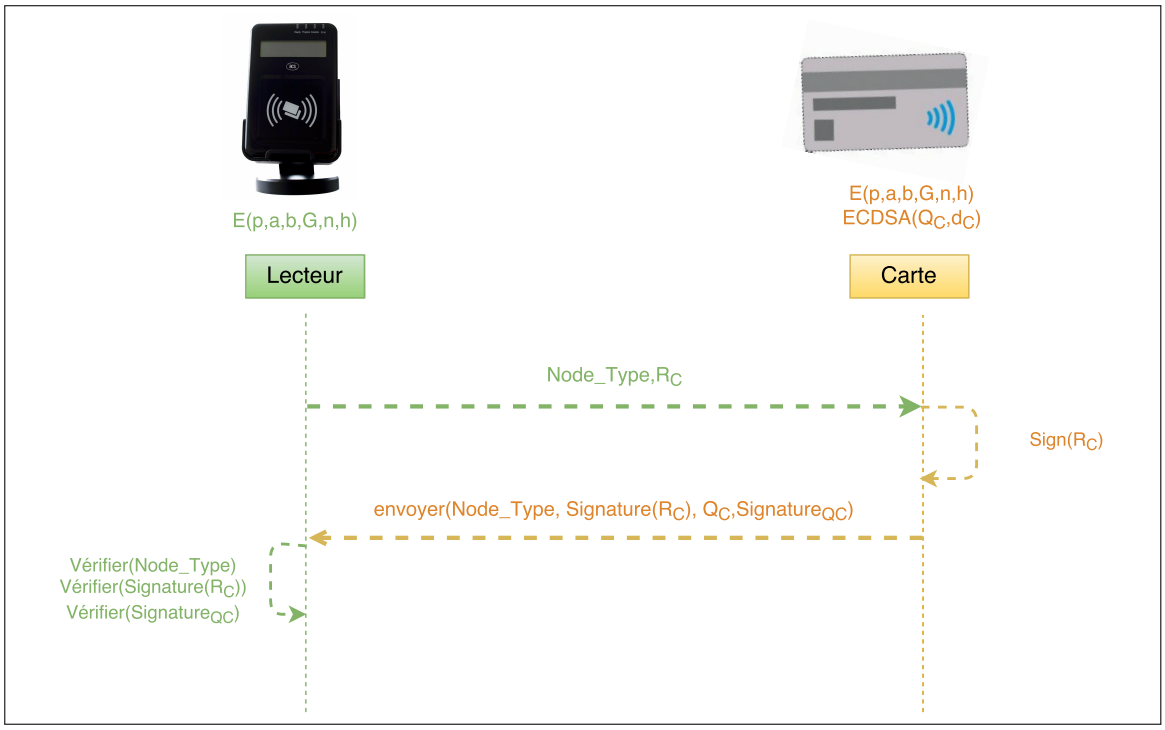

Figure 5.2 1ère variante du protocole : authentification de la carte

Nous avons varié les paramètres de courbes elliptiques pour tester l'efficacité en termes de temps d'exécution d'authentification de la carte. La figure 5.3 permet de constater que les spécifications *NIST* P-192, P-224 et P-256 respectent la contrainte de temps d'exécution que nous avons imposée, soit au dessous de 1 s. En passant au paramètres P-384, l'opération d'authentification de carte double son temps d'exécution, et le triple en passant aux paramètres P-521. Nous constatons aussi que les temps de transmissions NFC ne dépassent pas 150 ms (143 ms pour les paramètre P-521) dans cette implémentation, soit en moyenne 10% du temps total d'exécution.

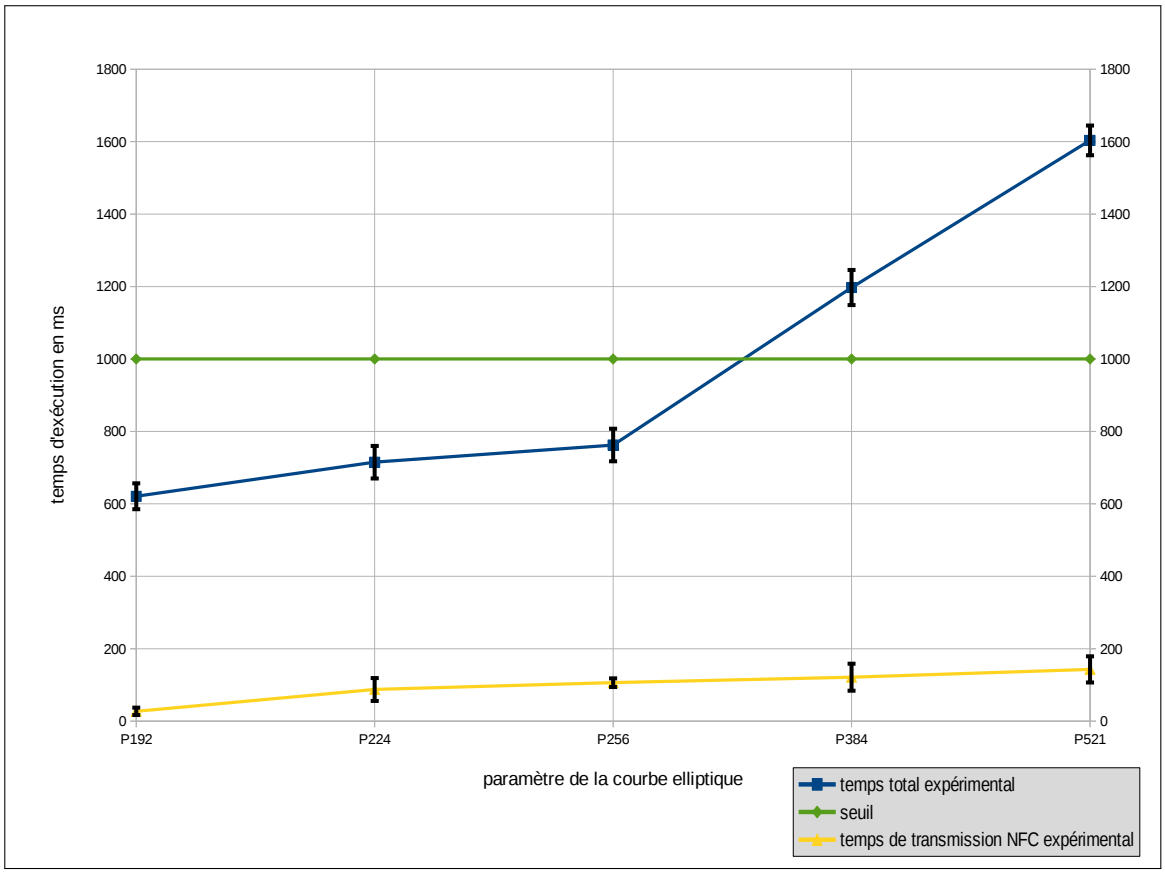

Figure 5.3 Temps d'exécution d'authentification de la carte en ms par la première variante du protocole en fonction de paramètres de la courbe elliptique, represéntés avec leurs intervalles de confiance à 90%.

5.4 2ème protocole : authentification mutuelle

Comme l'indique la figure 5.4, nous implémentons dans ce cas une authentification mutuelle d'une carte et d'un lecteur préalablement certifiés par la même unité de déploiement.

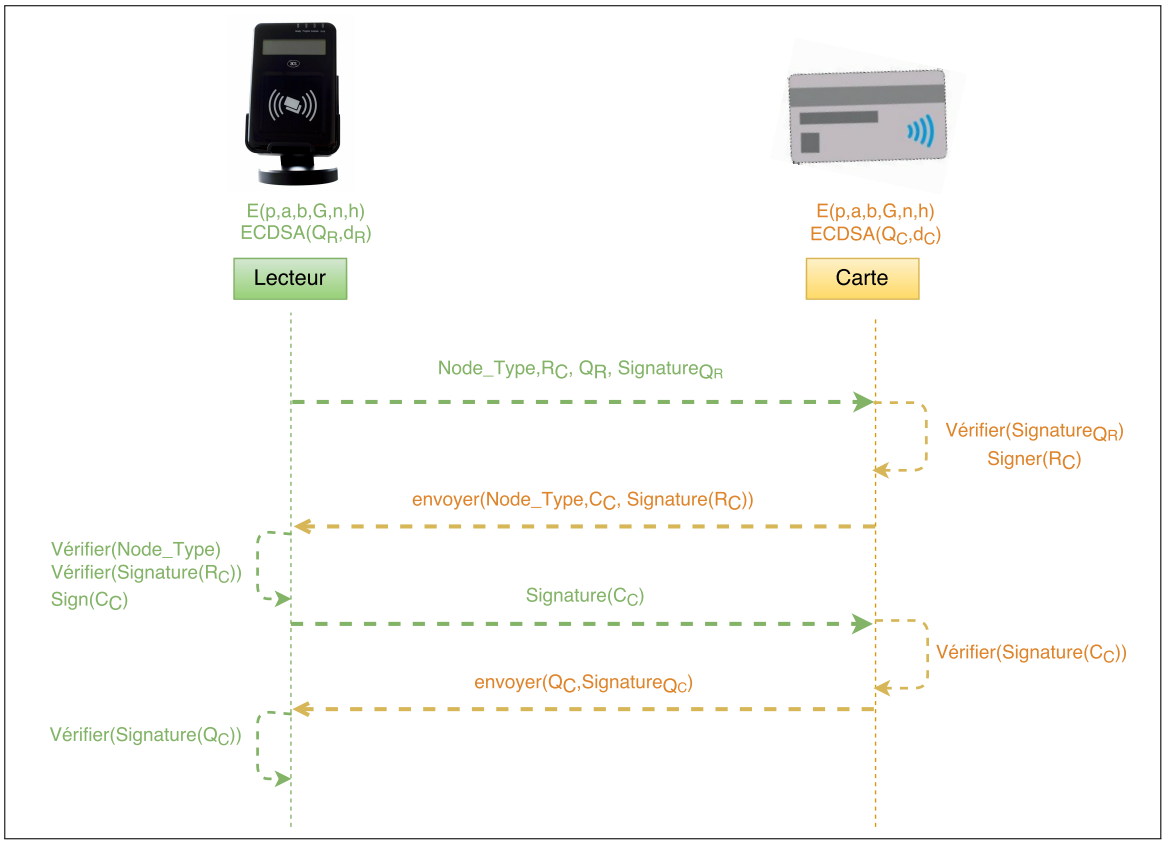

Figure 5.4 2ème variante du protocole : authentification mutuelle

La performance de notre solution en termes de temps d'exécution est naturellement affectée puisque nous ajoutons une communication NFC ainsi qu'un calcul de vérification du certificat du lecteur. Cependant, tel que nous pouvons le constater à la figure 5.5, nous considérons que ce ralentissement d'exécution n'est pas très significatif. Une différence est appréciable seulement pour P-521 (plus de 200 ms) et est négligeable (moins de 100 ms) pour tous les autres paramètres. En effet, les paramètres de courbe elliptique qui respectent la contrainte de temps d'exécution restent les mêmes. Nous pouvons donc utiliser les mêmes paramètres P-192, P-224 et P-256 sans dépasser 1 s, et sans ralentir la transaction de façon significative.

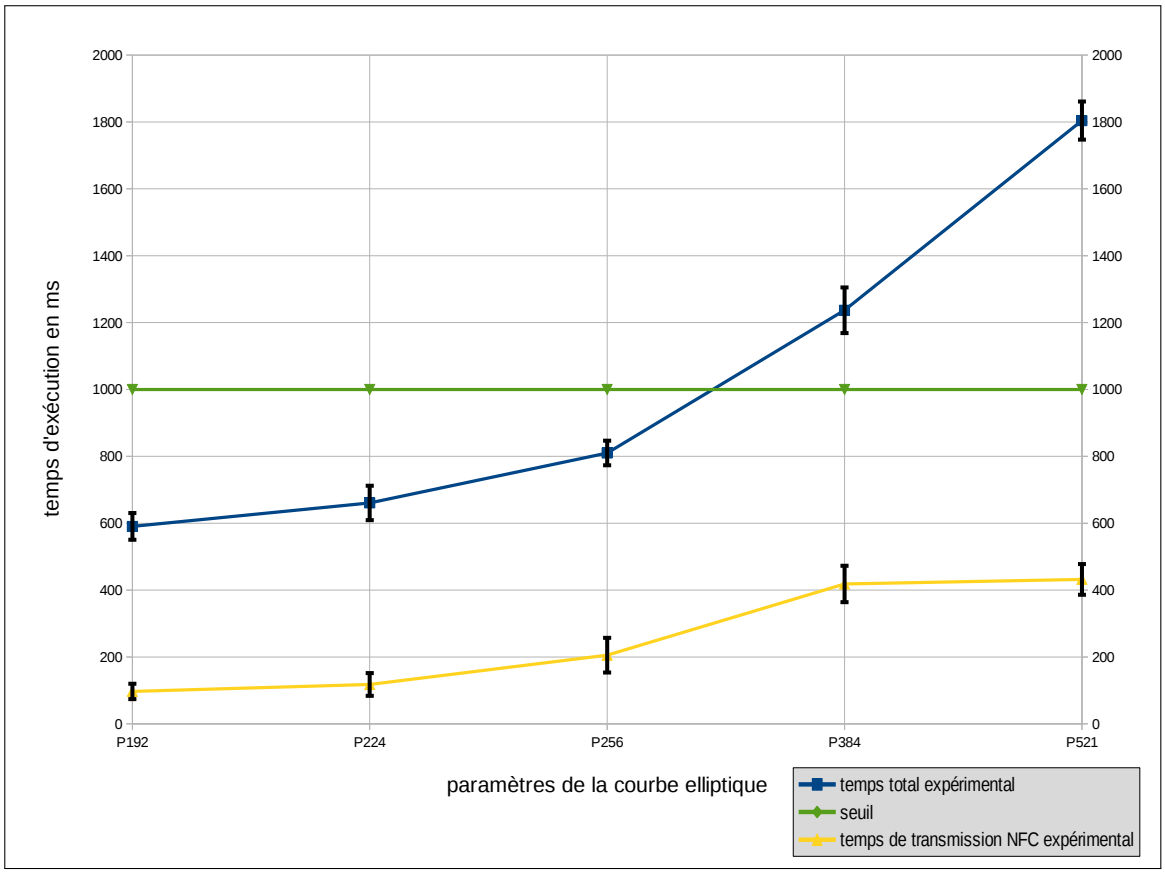

Figure 5.5 Temps d'exécution d'authentification de la carte en ms par la deuxième variante du protocole en fonction de paramètres de la courbe elliptique, representés avec leurs intervalles de confiance à 90%.

5.5 3ème protocole : authentification mutuelle, canal sécurisé pour envoyer le UID

Ce protocole implémente une authentification mutuelle tel que décrit plus haut et un chiffrement de session établi par un échange de clé cryptographique ECDH. La figure 5.6 illustre les échanges entre le lecteur et la carte pour effectuer une authentification mutuelle et un établissement de canal sécurisé. La clé échangée est utilisée pour chiffrer les données sensibles à échanger entre le lecteur et la carte telles que le UID. Dans cette implémentation du protocole, nous avons opté pour un chiffrement *Advanced Encryption Standard* (AES) pour envoyer le UID.

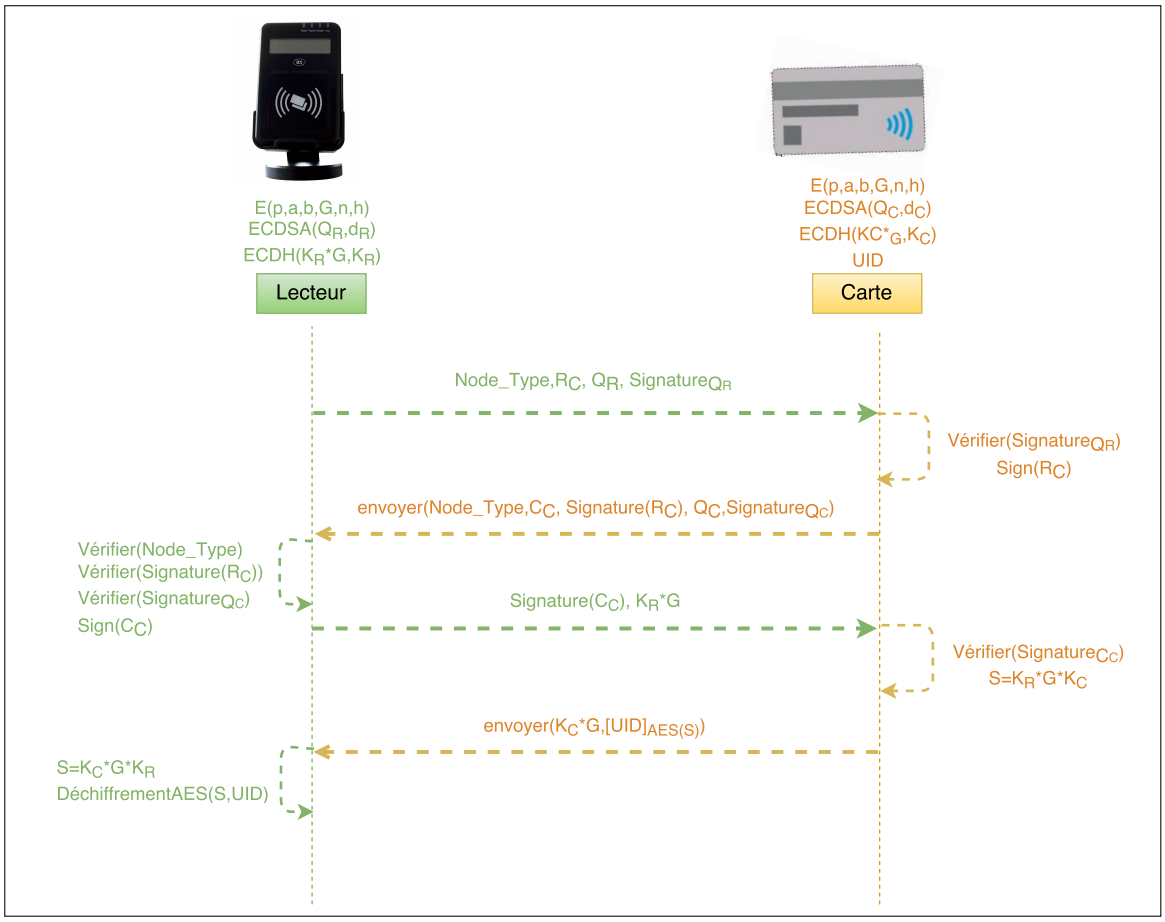

Figure 5.6 3ème variante du protocole : authentification mutuelle et établissement de canal sécurisé avec l'algorithme d'échange de clés ECDH

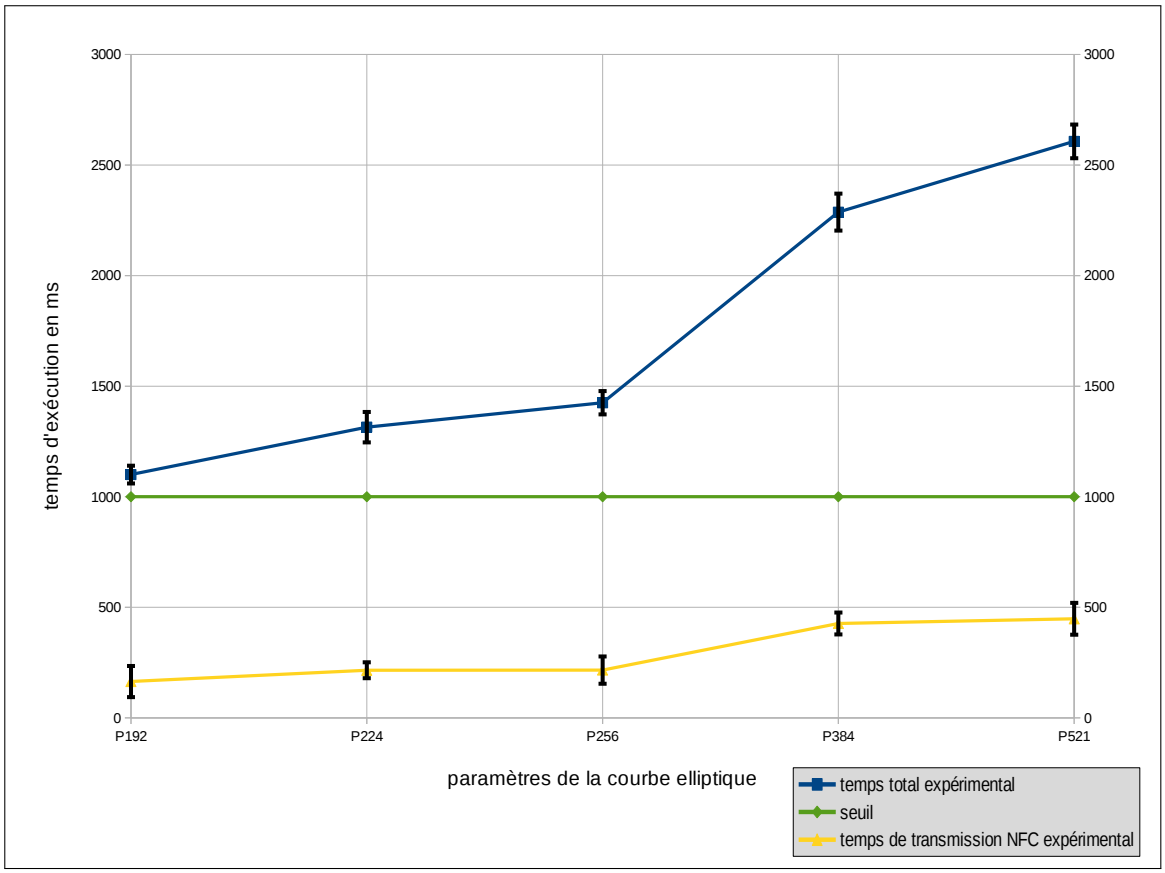

Figure 5.7 Temps d'exécution d'authentification de la carte en ms par la troisième variante du protocole en fonction de paramètres de la courbe elliptique, representés avec leurs intervalles de confiance à 90%.

Comme nous pouvons le constater à la figure 5.7, les temps de calcul sont nettement plus longs et dépassent la contrainte d'exécution de 1 s que nous avons fixée. En effet, la phase d'établissement de canal sécurisé et de transfert de UID est assez coûteuse en termes de calcul. Pour cela, nous avons essayé d'améliorer la performance d'exécution en variant les paramètres de courbe elliptique pour ECDH pour une même courbe ECDSA. Nous testons par exemple une courbe ECDSA P-256 avec des paramètres de courbes ECDH P-192 et P-224. Nous constatons dans le tableau 5.4 que la variation des paramètres de courbe de l'algorithme ECDH nous a permis d'améliorer le temps d'exécution mais ne nous a pas malheureusement permis d'attendre le niveau de performance désiré de 1 s.

Tableau 5.4 Temps d'exécution de la troisième version du protocole avec une courbe P-256 pour la signature ECDSA et en variant les paramètres de la courbe elliptique de l'algorithme ECDH en ms (sur un Nexus 5).

Opération	P-192	P-224	P-256
Temps d'exécution	1385,75	1400,05	1424,95

Nous constatons aussi que les temps de transmission NFC sont assez courts comparativement au temps total d'exécution, ce qui appui notre hypothèse du temps de calcul ECDH coûteux. Cependant, ce temps de calcul petit nous donne plus d'espoir de possibilité d'amélioration de performance de cette version de protocole pour atteindre des temps d'exécution acceptables. En effet, une des piste d'amélioration sera d'utiliser d'autres spécifications de courbes elliptiques.

5.6 Discussion

Nous avons dans ce chapitre présenté nos résultats reliés à l'évaluation de viabilité en termes de temps d'exécution de la solution d'authentification mobile NFC présentée dans le chapitre 4. Cette étude nous a permis d'identifier la variante du protocole ainsi que les paramètres de courbes elliptique les mieux adaptés à une solution assurant la fluidité du contrôle d'accès. D'une part l'authentification mutuelle est à peine plus coûteuse que l'authentification simple de la carte, et offre des protections supplémentaires en termes de prévention d'attaque de clonage par force brute, telle que celles que nous avons décrites au chapitre 2. En ce qui concerne le choix de courbes elliptiques, les paramètres NIST P-192, P-224 et P-256 nous permettent de rester sous le seuil d'un temps idéal de 1 s pour compléter la transaction complète. Comme nous avons pu modéliser et vérifier que le temps de calcul domine amplement le temps total, il est possible d'espérer qu'avec du matériel plus performant, par exemple avec des SE matériel spécialement adaptés pour ce type de calcul cryptographique, il serait possible de réduire le temps de calcul en dessous du seuil d'une seconde pour les autres paramètres NIST plus sécuritaires aussi.

Finalement, nous avons constaté que l'utilisation d'un protocole d'échange de clé ECDH pour chiffrer des informations sensibles transmises par NFC est trop coûteux, quels que soient les paramètres de courbe elliptique choisis. Ceci dit, la confidentialité des informations transmises n'est pas un pré-requis des applications d'authentification. Normalement, dans les normes RFID et NFC, l'UID est un simple identificateur de carte, équivalent à une adresse MAC pour une carte réseau ; il n'est donc pas confidentiel. Cependant, cet identificateur est souvent utilisé par des systèmes de contrôle d'accès physique comme identifiant unique de l'usager. Cet identifiant est retransmis par le lecteur au contrôleur qui le compare à une base

de données afin de décider si donner l'accès ou non. Comme l'interface de communication entre le lecteur et le contrôleur n'est pas authentifiée, la sécurité du système dépend de la confidentialité du UID. Une situation similaire se présente dans les systèmes de paiement par carte de crédit où le numéro de carte de crédit est transmis en clair par RFID ou NFC. Dans les deux cas, il s'agit d'un faux problème, car si les communications entre le lecteur (ou terminal de paiement) et le contrôleur (banque) étaient authentifiés, la carte n'aurait pas besoin de transmettre aucune information confidentielle au lecteur ; elle aurait seulement besoin de s'authentifier. Le problème constaté avec l'utilisation de protocole d'échange de clé ne se poserait même plus. Dans le monde du contrôle d'accès de nouvelles normes telle que le *Open Supervised Device Protocol* (OSDP) et les modes futurs de paiement RFID/NFC prévoient en effet de franchir cette étape. Dans l'intérim, il existera donc un compromis entre confidentialité du UID (ou numéro de carte de crédit) et performance.

CHAPITRE 6

CONCLUSION

L'adaptation de la technologie communication à champ raproché (NFC) au plateformes mobiles promet de révolutionner les services sans contact de part de leur convivialité, facilité et rapidité d'utilisation. Quoique que cette adaptation offre le potentiel d'adresser certaines des failles de sécurité inhérentes à la technologie identification par radio fréquence (RFID) qui l'a précédée, ce passage aux plateformes mobiles engendre de nouveaux facteurs de risque liés notamment à la nature de l'environnement d'exécution de ces applications, soit le système d'opération mobile, qui héberge aussi d'autres applications tels que des jeux, des réseaux sociaux et d'autres applications de divertissement. Cet environnement d'exécution peut héberger aussi des applications malveillantes menaçant l'intégrité des données de ces services mobiles NFC. Nous avons donc, dans ce mémoire, répertorié et évalué de façon qualitative les différentes solutions possibles pour minimiser l'impact de telles menaces dans le contexte d'utilisation de plateformes mobiles dans des applications d'authentification par NFC, notamment l'utilisation de *Secure Element* (SE) pour assurer le stockage sécuritaire de clés cryptographiques.

Cependant, notre contribution la plus importante est d'avoir proposé un protocole d'authentification mutuelle basé sur une infrastructure à clé publique (ICP) en utilisant la cryptographie à courbe elliptique (ECC), de façon à tenter de palier aux problèmes inhérents de la technologie RFID et des propositions initiales d'authentification par NFC. Ensuite, nous avons évalué les performances de cette solution en termes de temps d'exécution de différentes variantes du protocole proposé. Cette évaluation nous a permis d'identifier les paramètres de courbe elliptique qui respectent les contraintes de temps d'exécution, ainsi que les modes d'utilisation de ces protocoles qui seraient viables dans un contexte d'authentification.

Au cours de l'élaboration et l'implémentation de ce protocole, le premier problème rencontré a été le choix de solution de stockage de clés cryptographiques. Plusieurs solutions d'exécution et de stockage matériel dites SE sont disponibles sur le système d'exploitation mobile Android. Ces solutions purement matérielles constituent un environnement isolé assurant le niveau de sécurité nécessaire pour des applications manipulant des données sensibles telles que le paiement mobile. Cependant, ces solutions sont complexes, difficiles à déployer et impliquent un partenariat avec les fournisseurs de ces éléments matériel de sécurité. L'apparition de la nouvelle fonctionnalité *Host Card Emulation* (HCE) sur Android a facilité le déploiement de nouveaux services sans contact en permettant l'émulation logicielle de carte

NFC. Nous avons dans ce contexte détaillé les différentes solutions d'émulation de carte NFC sur Android ainsi que les options de stockage de clés cryptographiques. Dans le cadre de notre étude, nous n'avons pas eu l'occasion de tester une solution d'environnement matériel d'exécution et de stockage de données matérielle étant que ces environnements ne sont pas encore ouverts aux développeurs.

De plus, l'étude expérimentale a montré que nous ne pouvons pas rester fidèle à la solution d'authentification proposée au chapitre 4, puisque cette solution ne respecte pas les contraintes de temps d'exécution que nous avons imposés.

Pour cela, nous envisageons d'améliorer les performances d'exécution du protocole, notamment de calcul cryptographique de signature *Elliptic Curve Digital Signature Algorithm* (ECDSA) et d'échange de clé *Elliptic Curve Diffie-Hellman* (ECDH). Nous souhaitions donc réaliser une implémentation javacard de ce protocole, dans laquelle un composant matériel tel que la puce NFC de la famille SmartMX de la compagnie NXP, aurait joué le rôle du lecteur, ou nous pourrons utiliser d'accélérateur cryptographique matériel spécialement conçu afin d'obtenir de meilleures performances de calcul cryptographique.

RÉFÉRENCES

ALIMI, V. (2012). *Contributions to the deployment of mobile services and to the analysis of transactions security*. Thèse de doctorat, Université de Caen.

AOSP (2014). Near field communication. `http://developer.android.com/guide/topics/connectivity/nfc/index.html/`. Consulté le 12 janvier, 2014.

BRUN-MUROL, P. (2012). *Vers une méthodologie normalisée d'évaluation des solutions RFID en application de sécurité*. Mémoire de maîtrise, Ecole Polytechnique de Montréal.

EMMS, M., ARIEF, B., FREITAS, L., HANNON, J. et VAN MOORSEL, A. (2014). Harvesting high value foreign currency transactions from EMV contactless credit cards without the PIN. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 716–726.

HADHIRI, A. (2012). *Sécurité des applications Android : menaces et contremesures*. Mémoire de maîtrise, École de Technologie Supérieure.

HANCKE, G. P. (2006). Practical attacks on proximity identification systems (short paper). *2006 IEEE Symposium on Security and Privacy (S&P 2006), 21-24 May 2006, Berkeley, California, USA*. 328–333.

KARSTEN NOHL, H. P. (2007). Mifare, little security despite obscurity. *Chaos Computer Congress*.

KHANDAVILLI, A. P. (2012). *A mobile based access control system using identity based encryption with non-interactive zero knowledge proof of authentication*. Mémoire de maîtrise, Dalhousie University.

KIRSCHENBAUM, I. et WOOL, A. (2006). How to build a low-cost, extended-range rfid skimmer. Cryptology ePrint Archive, Report 2006/054. `http://eprint.iacr.org/`.

LIAO, Y.-P. et HSIAO, C.-M. (2014). A secure ECC-based RFID authentication scheme integrated with id-verifier transfer protocol. *Ad Hoc Networks*, 18, pp. 133–146.

MARKANTONAKIS, K. (2012). Practical relay attack on contactless transactions by using nfc mobile phones. *Radio Frequency Identification System Security : RFIDsec*, 12, 21.

MILOSCH MERIAC, H. P. (2010). Analyzing a modern cryptographic RFID system HID iClass demystified. *Chaos Computer Congress*.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (2001). FIPS 197. *National Institute of Standards and Technology, November*, 1–51.

OWASP (2014). Owasp mobile security project. `https://www.owasp.org/index.php/OWASP_Mobile_Security_Project/`. Consulté le 9 aout, 2015.

POCATILU, P. (2011). Android applications security. *Informatica Economica*, 15, 163 – 71.

THEVENON, P.-H. (2011). *Sécurisation de la couche physique des communications sans contact de type RFID et NFC*. Thèse de doctorat, École Doctorale d'Electronique, Electrotechnique, Automatique & Traitement du Signal (EEATS).